Studies on Advanced Metaheuristics employing Local Clustering for Job-shop Scheduling Problem

Yasumasa Tamura

Submitted in partial fulfillment of the requirements

for Degree of Doctor of Information Science

Hokkaido University

February 2015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Research Objective

Combinatorial optimization is a kind of optimization problems where the feasible solutions are composed of the combinatorial discrete structure such as permutation or assignment. The combinatorial optimization is one of the most fundamental problem to model the assignment problems, the routing problems or the scheduling problems in the real world. In general, many of combinatorial optimization problems have the multimodal objective function and the number of feasible solution increases exponentially in accordance with the increase of problem size. From these characteristics, many studies try to solve the combinatorial optimization problems with the approximate algorithms, such as heuristics or metaheuristics.

Job-shop scheduling problem (JSP) is a combinatorial optimization prob-

lem to determine a feasible and efficient schedule to process multiple jobs on multiple machines [Fre82]. The JSP consists of a set of multiple jobs (tasks) and a set of multiple machines (resources) and the objective of the JSP is to determine an ideal or efficient schedule to process each job on each machine. Because many scheduling problems in the real world are based on the JSP, it has been studied as an important subject in the field of the operations research, the computer science and so on [CB76, ZF94]. The computational complexity theory classifies the JSP into the NP-hard problem class as well as the other many combinatorial optimization problems. In addition, the JSP is regarded as more difficult problem than the *travelling salesman problem* (TSP), a typical combinatorial optimization problem, by the comparison with the *flow-shop scheduling problem* (FSP), a kind of the restricted JSP [GJS76, GJ79]. It is actually known that the general fitness landscape formed by the objective function of the JSP has strong multimodality.

The classical solution algorithms for the JSP are based on a typical exact algorithm named *branch and bound method*. B.Giffler and G.L.Thompson shows the procedure to enumerate the restricted feasible schedules, which is called the *GT method* [GT60], and Brooks and White proposes the branch and bound method based on the GT method [BW69]. Because the branch and bound method is the exact algorithm, it is guaranteed that the method obtains the optimal solution. On the other hand, due to the complexity of the JSP, the branch and bound method requires impractical computational time in many cases. For that reason, although several improved algorithms

based on the branch and bound method had been proposed, most of the practical solution methods are based on the approximate algorithm such as heuristics or metaheuristics.

Adams et al. proposes the *shifting bottleneck procedure* (SB) for the JSP in 1988 [ABZ88]. SB is known as a forerunner of the approximate algorithms for the JSP and one of the typical heuristic algorithm for the JSP. Also the *dispatching priority rules* are known as the effective and practical greedy heuristics for the JSP [PI77]. These algorithms are often used to determine a good schedule in the manufacturing systems, since they can find near-optimal or moderately good solutions in the reasonable computational time.

In accordance with the dramatic development of the computer performance in the recent decades, the effectiveness of the metaheuristics for the JSP is reported by many studies. In particular, *local search* (also called *neighborhood search*) with the *critical paths* or the *critical blocks* are known as the effective solution methods for the JSP. The critical path is defined as a subsequence of successive operations whose sequence directly affects the makespan [Bal69] and is known as one of the most effective heuristics for the JSP. Several metaheuristics employing local search based on the neighborhood structure using the critical paths are proposed and verified their effectiveness [LAL92, Tai94, NS96, NS05]. They also showed that the neighborhood structure using the critical paths can be used to search for the near-optimal solutions efficiently. These algorithms are very effective to solve the JSP, however, it is also reported that the neighborhood structure based

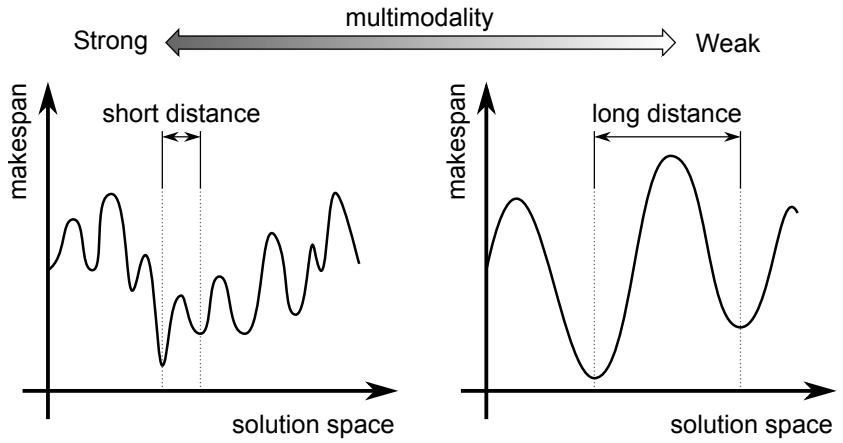


Figure 1.1: Multimodality and distance between local optima

on the critical paths requires highly complex mechanisms to generate effective neighborhood solutions [WHW05]. As the other effective metaheuristics, Furukawa et al. proposes *local clustering organization* (LCO) which is a probabilistic metaheuristics employing local search for the combinatorial optimizations inspired by *self-organizing map* (SOM) proposed by Kohonen [Koh98]. The effectiveness of LCO for solving the JSP is verified in comparison to genetic algorithm (GA), one of the typical metaheuristics. Although LCO searches for the solutions without the critical paths, its performance is competitive with the metaheuristics using the critical paths in some cases. On the other hand, it has been suggested the search performed by LCO is sometimes trapped in the local optima.

This thesis focuses on the effective neighborhood structure to escape from local optima in LCO. Though the multimodality of the JSP is very strong, the strength of multimodality is different depending on the problem structure.

On JSP, it is known that the strength of the multimodality depends on the relationship between the number of jobs and the number of machines. This thesis discusses the effective strategies to escape from local optima particularly by focusing the strength of the multimodality. In general, it is assumed that the multimodality causes the difference of distance between local optima as shown in Figure 1.1. It also causes the difference of effective strategies to escape from local optima. As a general knowledge for the combinatorial optimizations, this thesis proves the effectiveness of the strategy for escaping from local optima depends on the strength of the multimodality.

This thesis also focuses on the *dynamic scheduling*. The scheduling problem are generally divided into two kinds of problem sets, the *static scheduling* and dynamic scheduling. The static scheduling deals with the predetermination of the schedule. In the static scheduling, all of information to determine the schedules are given in advance. Also the accuracy of the schedule obtained in a practical computation time is emphasized in it. On the other hand, the dynamic scheduling deals with real time construction of the schedules or modification of the predetermined schedules. In particular, the modification of the predetermined schedules weighs with keeping the production efficiency high in the manufacturing systems. In the dynamic scheduling, the fast response of the solution algorithms is emphasized for fast modification of the schedule. In this thesis, the effectiveness of the metaheuristics employing local search is discussed on the basis of the solution method for schedule modifications.

Table 1.1: An instance of 4×3 problem

—	1	2	3
J_1	(M_1, 11)	(M_2, 9)	(M_3, 16)
J_2	(M_2, 25)	(M_1, 10)	(M_3, 13)
J_3	(M_2, 9)	(M_3, 11)	(M_1, 15)
J_4	(M_3, 12)	(M_2, 14)	(M_1, 11)

1.2 Job-shop Scheduling Problem (JSP)

1.2.1 Description of the Problem

The JSP is generally described by a set of multiple jobs $J = \{J_1, J_2, \cdots, J_n\}$ and a set of multiple machines $M = \{M_1, M_2, \cdots, M_m\}$, where the notation n and m denote the number of jobs and the number of machines. Each job consists of m operations $J_i = (o_{i,1}, o_{i,2}, \cdots, o_{i,m})$, where $o_{i,k}$ corresponds to the k-th process of J_i. The order of the operations for each job is respectively given as a predetermined technological sequence of machines. The processing time $p_{i,k}$ for each operation $o_{i,k}$ is also given in advance. Table 1.1 shows an instance of the JSP. In Table 1.1, for example, $(J_1, 1) = (M_1, 11)$ denotes that the operation $o_{1,1}$ is processed on the machine M_1 with the processing time 11.

To describe the JSP, this thesis provides following notations.

$s_{i,k}$: the start time of the operation $o_{i,k}$
$c_{i,k}$: the completion time of the operation $o_{i,k}$
$r_{i,k}$: the machine on which the operation $o_{i,k}$ is processed

Owing to the definition of the JSP, they are rightly restricted as follows;

$$s_{i,k} \geq 0 \tag{1.1}$$

$$c_{i,k} \geq 0 \tag{1.2}$$

$$r_{i,k} \in \{1, 2, \cdots, m\} \tag{1.3}$$

The objective of the JSP is generally defined as the minimization of the *makespan*, the total (maximum) processing time to process all of the jobs, which described in Equation 1.4.

$$\text{minimize} \quad \max_i \{c_{i,m}\} \tag{1.4}$$

The JSP also has several constraints as follows.

1. Every operation is not interrupted and resumed in processing.
2. Each operation $o_{i,k+1}$ cannot be started processing before the previous operation $o_{i,k}$ is not completed.
3. While a job is processed by a machine, the job is not processed by any other machines simultaneously.
4. While a machine processes a job, any other jobs are not set to the same machine simultaneously.

From the constraint 1, the completion time $c_{i,k}$ is determined by $s_{i,k}$ and $p_{i,k}$ as Equation 1.5.

$$c_{i,k} = s_{i,k} + p_{i,k} \tag{1.5}$$

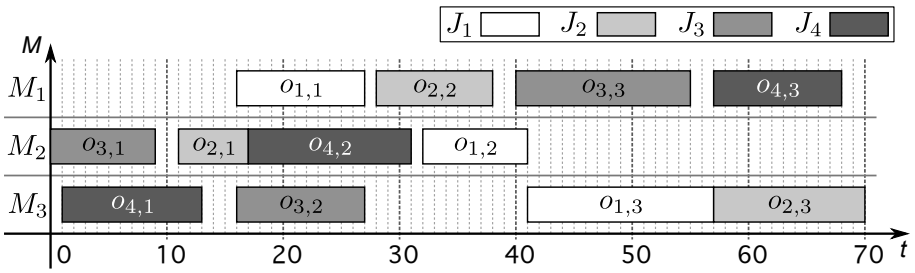


Figure 1.2: A feasible schedule

From the constraints 2 and 3, the start time $s_{i,k}$ is restricted by the completion time of the previous operation as shown in Equation 1.6.

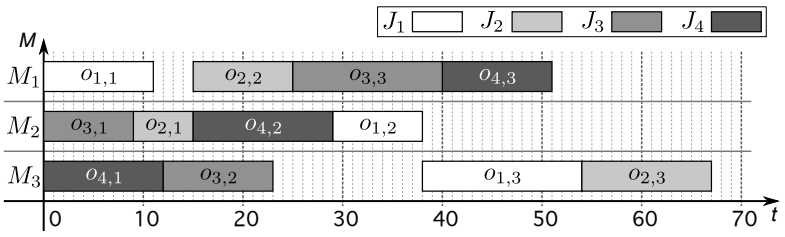
$$s_{i,k+1} \geq c_{i,k} \tag{1.6}$$

From the constraint 4, the start time $s_{i,k}$ is also restricted the completion time of the other operations processed on the machine $r_{i,k}$. The constraint 4 is described as Equation 1.7, where the notation $c_{i',k'}$ denotes the completion time of another operation $o_{i',k'}$ and the equation $o_{i',k'} \prec o_{i,k}$ denotes the priority of $o_{i',k'}$ over $o_{i,k}$.

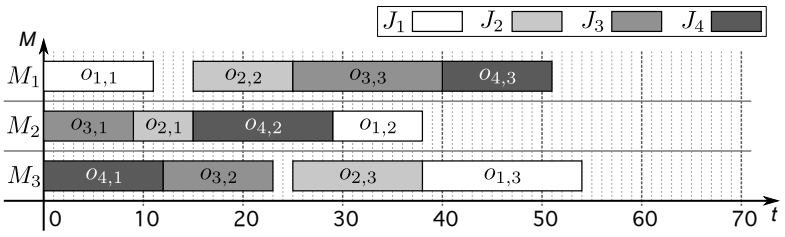
$$s_{i,k} \geq c_{i',k'} \quad (r_{i,k} = r_{i',k'}, o_{i',k'} \prec o_{i,k}) \tag{1.7}$$

To determine the start time $s_{i,k}$, the priority relations among the operations should be decided. In other words, the priority relations such as $o_{i',k'} \prec o_{i,k}$ are the decision variables in the JSP.

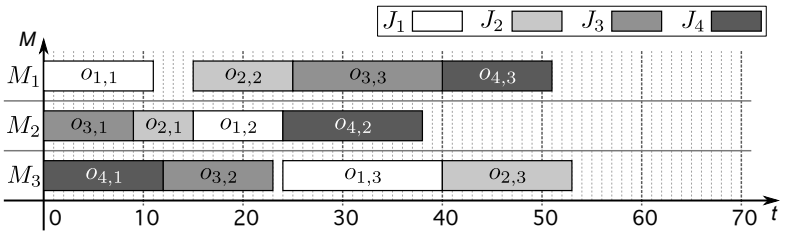
Figure 1.2 shows one of the feasible schedules for the instance given by Table 1.1. The chart in Figure 1.2 is called *Gantt chart*.



(a) A semi-active schedule



(b) An active schedule



(c) An optimal schedule

Figure 1.3: Subset of the feasible schedule

1.2.2 Search Space for the JSP

The feasible region of the JSP is defined as a set of schedules which satisfy all of the constraints. In addition, two important subsets, *semi-active schedule* and *active schedule*, are defined in the feasible region [CMM67, Bak74, RK76]. The definition of each subset is explained as follows.

Semi-active schedule

A set of schedules which are feasible and have no operation can be started earlier without rearranging sequences of jobs on any machines. An example of the semi-active schedule is shown in Figure 1.3(a). It is proved that an optional feasible schedule can be changed into a semi-active schedule by cutting idle time down on each machine. For example, the feasible schedule shown in Figure 1.2 is changed into the semi-active schedule shown in Figure 1.3(a). In Figure 1.3(a), when the order of $o_{2,3}$ and $o_{1,3}$ is swapped, the schedule is changed into another schedule shown in Figure 1.3(b) which is one of the active schedule. It is also proved that an optional semi-active schedule can be changed into an active schedule when the sequence of proper operations are changed.

Active schedule

A set of schedules which are feasible and have no operation can be started earlier without delaying the start time for other operations even if the rearrangement of operations is allowed. It is important that the transition from an active schedule to another active schedule including the optimal schedule sometimes requires more search steps than that from a semi-active schedule. Figure 1.3(c) shows the optimal schedule for the instance shown in Table 1.1. The optimal schedule is closer to the semi-active schedule shown in Figure 1.3(a) than the active one Figure 1.3(b). It sometimes affects the effectiveness of local search.

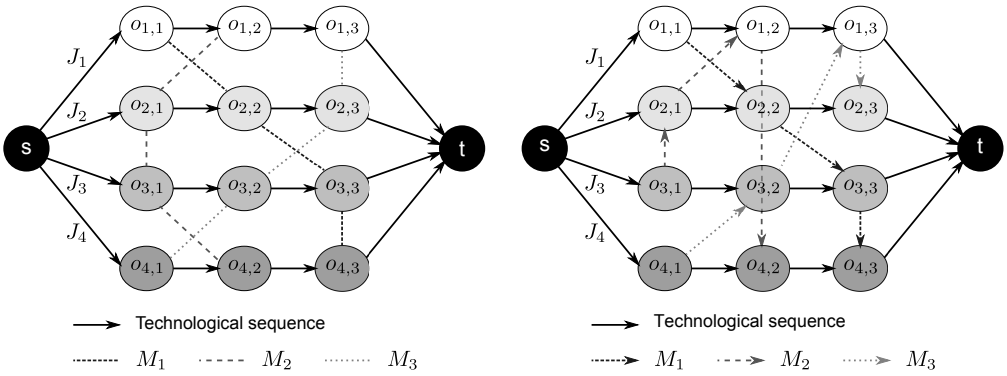
It is guaranteed that each subset contains the optimal solutions. Therefore, most of solution algorithms search for the limited schedules in the subsets to avoid wasteful searches. The solution algorithms requires $O(nm)$ computational steps to create a semi-active schedule, and they require no extra mechanism to limit the search space because any feasible schedules can be converted into semi-active schedules. On the other hand, the solution algorithms require extra mechanisms to limit the search space to the active schedule. The *GT method* proposed by Giffler and Thompson is typical algorithm to limit the search space in the active schedule [GT60]. The creation of an active schedule using GT method requires $O(n^2m)$ computational steps.

1.2.3 Representation of Schedules

The following two solution representations, called the *disjunctive graph* and the *permutation with repetition*, are generally used in metaheuristics for the JSP. The disjunctive graph is also a typical way to describe the constraints of the JSP. While it can represent a schedule directly, the permutation with repetition indirectly represents a schedule.

Disjunctive graph

The set of nodes in the disjunctive graph consists of the operations and two dummy nodes, s and t. The nodes correspond to each operation (the operation nodes) are connected by directed edges which denote the technological sequence in each job. The dummy nodes s and t are also connected



(a) Disjunctive graph for Table 1.1 (b) Conjunctive graph for Figure 1.3(c)

Figure 1.4: Solution representation: disjunctive graph

to the operation nodes by directed edges to denote the head and tail of the technological sequence. In addition, the operation nodes are connected by undirected edges, named the *disjunctive arcs*. The disjunctive arcs denote the machine restriction of each operation. Figure 1.4(a) shows the disjunctive graph for the instance shown in Table 1.1. In Figure 1.4(a), the solid arrows denote the technological sequence and the disjunctive arcs are shown by the broken edges. A feasible schedule is represented using the graph by determining the operation sequence on each machine. The operation sequence on each machine is represented by several directed arrows called *conjunctive arcs*. Because the machine restriction of each operation is represented by the disjunctive arcs, when the graph represents a feasible schedule, the operations connected by the disjunctive arcs are connected by the conjunctive arcs. The graph with the conjunctive arcs is generally called the *conjunc-*

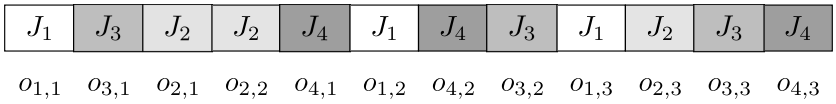


Figure 1.5: Solution representation: permutation with repetition

tive graph. Figure 1.4(b) shows the conjunctive graph corresponds to the schedule shown in Figure 1.3(c), where the broken arrows denote the conjunctive arcs. To represent a feasible schedule, the conjunctive graph must be a directed acyclic graph (DAG). In other words, when the conjunctive arcs are selected to create a DAG, the conjunctive graph certainly represents a feasible schedule.

Permutation with repetition

As a generalized permutation approach to represent a solution for the JSP, the solution representation based on the permutation with repetition is originally proposed by Bierwirth [Bie95]. According to the definition of the representation, a solution consists of the sequence of jobs, where each job appears m times repeatedly. On the solution representation, scanning it from left end to right end, the k^{th} occurrence of the job J_i corresponds to the operation $o_{i,k}$. For example, the permutation with repetition shown in Figure 1.5 represents the solution shown in Figure 1.3(c)

The sequence of operations can also be used as a solution representation. However, a solution represented by the sequence of operations often corre-

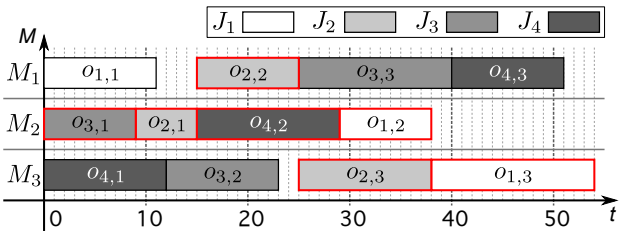


Figure 1.6: Critical path

sponds to infeasible schedules because it is not guaranteed that the sequence of operations follows the technological sequence. It causes inefficiency of searches due to the requirement of the mechanism to convert the solution into a feasible schedule such as the GT method. On the other hand, it is guaranteed that a solution represented by the permutation with repetition always corresponds to a feasible solution.

1.3 Metaheuristics employing Local Search for the JSP

In this section, we summarize the conventional algorithms for the JSP. Because this study particularly focuses on the metaheuristics employing local search, this section mainly summarizes the metaheuristics on the basis of the local search.

1.3.1 Local Search with Critical Paths

Local search is a typical approximate algorithm verified its effectiveness for many search problems including the combinatorial optimization. In general, the effectiveness of local search depends on the heuristics used to determine the neighborhood structure. The *critical path* is an effective heuristics in the JSP with the minimization of makespan.

The critical path is defined as a subsequence of successive operations whose sequence directly affects the makespan [Bal69]. Figure 1.6 shows an example of the critical path where the operations on the critical path are emphasized by red lines. To reduce inefficient searches, about seven kinds of the neighborhood structure, $N1$ – $N7$ operators, based on the critical path are proposed by several studies [DT93, BDP96, NS96, BV98, ZLGR07] and it is known that the effectiveness of the $N5$ operator is averagely higher than the others [WHW05]. These operators commonly prune the solutions which has the same makespan with the current solution from the search neighborhoods. Although this search mechanism to prune such solutions contributes to computation efficiency, there are several problems. For example, Figure 1.7 shows a certain case of the transition of the solution. When the current solution corresponds to B in the figure, to find the best solution F, the search algorithm has to allow the transition of the solution from B to the worse solution C or it has to find the solution E using multi-step neighborhood search. Because the multi-step neighborhood search mechanism generally tends to

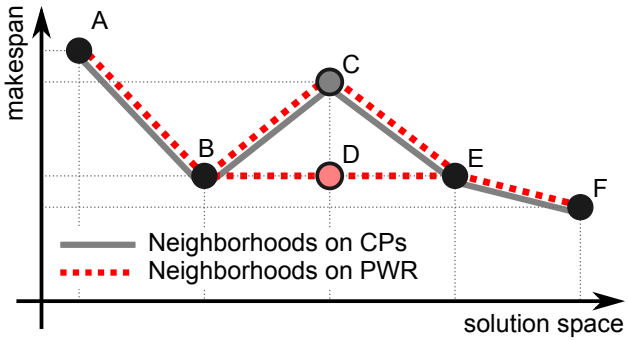


Figure 1.7: Transition of the solution

be complex, many effective algorithm for the JSP is based on metaheuristics such as *simulated annealing* [LAL92] or *tabu search* [Tai94, NS96]. The algorithms proposed especially in current decades are generally based on those metaheuristics and hybridized with iterated or multi-point search mechanism [PM00, NS05].

1.3.2 Local Clustering Organization

LCO is a probabilistic metaheuristic algorithm proposed by Furukawa et al. [FMW05]. The original idea of LCO is based on the Riccati-type learning equation which is used in Self-Organizing Map (SOM), a kind of neural networks proposed by Kohonen[Koh98]. The effectiveness of LCO for the JSP is also verified [FMW06, KS09].

LCO searches for an approximate solution by applying replacements repeatedly to a subset of consecutive entities, which is called *local* in LCO, in the current solution. The local is randomly selected in each searching step.

The replacements in the local are performed by *clustering methods*, which provides greedy searches. By repeating these methods, LCO realizes fast and effective local searches to find an approximate solution.

Here, we introduce some notations to explain LCO.

S_c : the current solution
d_c : the index of the center entity in selected local
$r(t)$: the clustering radius in t^{th} step

The general optimization (minimization) algorithm of LCO is described as follows and partly shown in Figure 1.8.

1. Set step $t \leftarrow 1$.
2. Initialize S_c at random.
3. Select an entity d_c in S_c at random. The local is selected as a subset of the consecutive entities which are within a radius $r(t)$ from the selected entity.
4. Select a clustering method stochastically.
5. Apply the selected clustering method to the selected local in S_c.
6. Replace t with $t + 1$.
7. If termination conditions of the algorithm are satisfied, stop the procedure. Otherwise, go back to step 3.

In this paper, we use four clustering methods, *SEM*, *IEM*, *SYM* and *SIM*. The additional notations are introduced to explain the clustering methods.

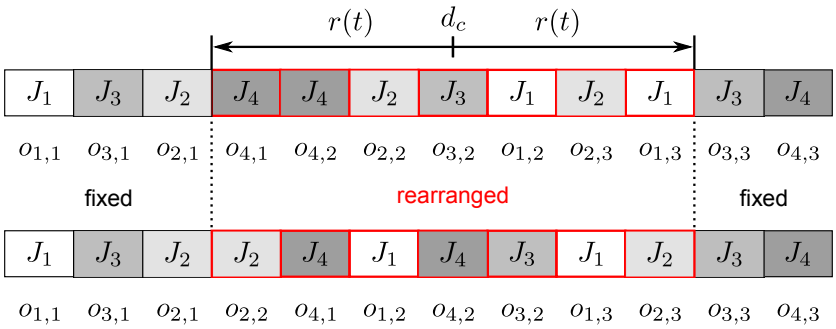


Figure 1.8: Local clustering

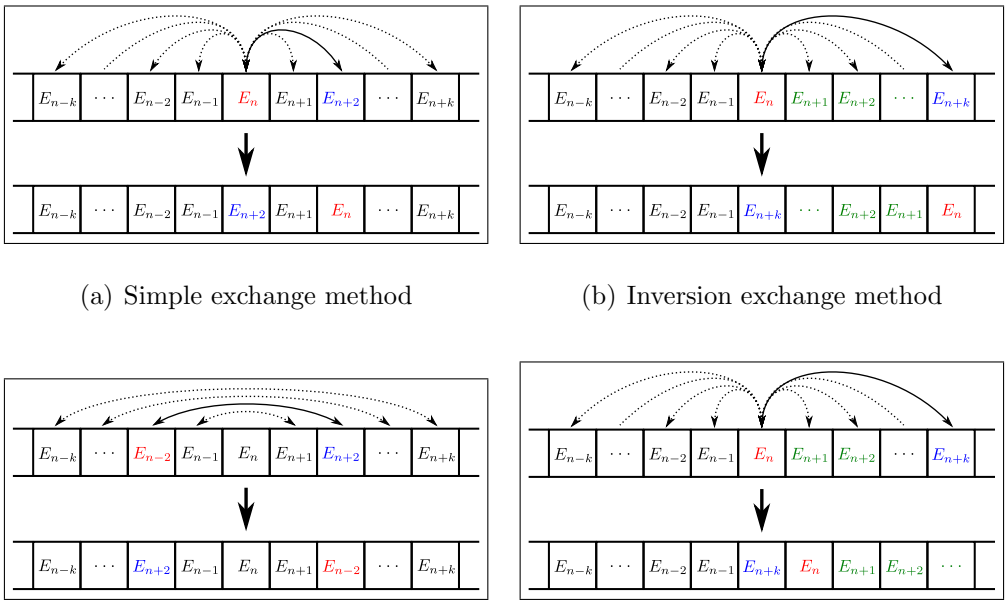
$S_c[d]$: the d^{th} entity in S_c
S_n : the candidate (neighbor) solution
$C(S_*)$: the cost of a solution S_*

In the following procedures, i is used for the variables in each clustering method, not for job numbers. Some examples of the replacement performed by each method are also shown in Figure 1.9, where E_d stands for the entity in the location $\mathbf{S_c}[d]$.

Simple exchange method (SEM)

This method generates S_n by exchanging two entities in the local of S_c. The procedure of SEM is described as follows;

1. Set $i = 1$.
2. Generate the new solution S_n by exchanging $S_c[d_c]$ for $S_c[d_c - i]$.
3. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.
4. Create S_n by exchanging $S_c[d_c]$ for $S_c[d_c + i]$.
5. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.



(a) Simple exchange method

(b) Inversion exchange method

(c) Symmetric exchange method

(d) Simple insertion method

Figure 1.9: Clustering methods ($1 \leq k \leq r(t)$)

6. Replace i with $i + 1$.

7. If $i \leq r(t)$, then go back to the step 2. Otherwise stop the procedure.

Inverse exchange method (IEM)

This method generates S_n by inverting the arrangement between two entities in the local of S_c. The procedure of IEM is explained as follows;

1. Set $i = 1$.

2. Generate the new solution S_n by inverting the subsequence between $S_c[d_c]$ and $S_c[d_c - i]$.

3. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.

4. Create S_n by inverting the subsequence between $S_c[d_c]$ and $S_c[d_c+i]$.

5. If $C(S_n) \leq C(S_c)$, replace S_c with S_n.

6. Replace i with $i+1$.

7. If $i \leq r(t)$, go back to the step 2. Otherwise stop the procedure.

Symmetric exchange method (SYM)

This method generates S_n by exchanging two entities in the local of S_c with a different procedure to SEM. The procedure of SYM is explained as follows;

1. Set $i = 1$.

2. Generate the new solution S_n by exchanging $S_c[d_c-i]$ and $S_c[d_c+i]$.

3. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.

4. Replace i with $i+1$.

5. If $i \leq r(t)$, go back to the step 2. Otherwise stop the procedure.

Simple insertion method (SIM)

This method gives the changes based on insertion to the local of S_c. The original implementation of SIM is proposed by Konno and Suzuki [KS09]. The procedure of SIM is explained as follows;

1. Set $i = 1$.

2. Generate the new solution S_n by removing $S_c[d_c - i]$ and inserting it in front of $S_c[d_c]$.

3. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.

4. Create a S_n by removing $S_c[d_c + i]$ and inserting it in front of $S_c[d_c]$.

5. If $C(S_n) \leq C(S_c)$, replace S_c with S_n.

6. Replace i with $i + 1$.

7. If $i \leq r(t)$, go back to the step 2. Otherwise stop the procedure.

In addition to the clustering methods, the clustering radius $r(t)$ has a great effect on accuracy and computational time of optimization. In LCO, solutions are iteratively searched by local clusterings. Since the clustering radius decides the partial solution size, if the radius is set to smaller one, the accuracy of the solution sometimes becomes worse because the widely covered optimization in the whole solution is not performed. On the other hand, the bigger radius causes the delay of computational time. From the above, the clustering radius is important to generate a solution which has highly approximate accuracy with fast computational time. Furukawa et al. suggest that Equation 1.8 is a relatively good setting in their paper proposing LCO originally.

$$r(t) = \frac{n \times m}{3} \tag{1.8}$$

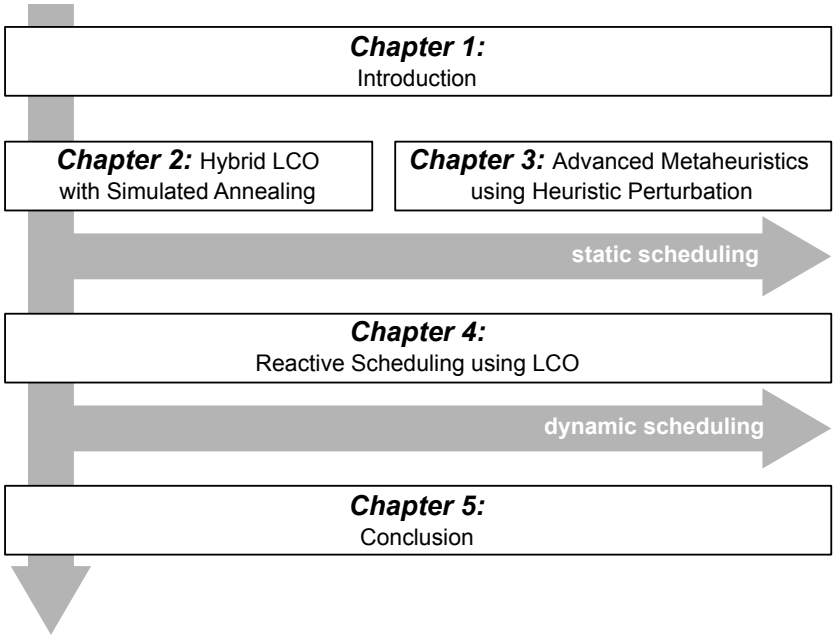


Figure 1.10: Outline of the thesis

1.4 Outline of the Thesis

This thesis is composed of 5 chapters shown in Figure 1.10 visually. In Chapter 2 and 3 discuss the effective mechanism to escape from local optima for the static problem. Chapter 2 particularly discusses the mechanism to escape from local optima for the square instances. Because the square instances have relatively strong multimodality, the mechanism proposed in Chapter 2 is based on simulated annealing to apply the random effect to the control mechanism for the transition of the neighborhood solutions. On the other hand, Chapter 3 discusses the mechanism for the rectangle instances which have relatively weak multimodality. When the fitness landscape has

weak multimodality, the distance between the local optima tends to be long. Therefore, Chapter 3 proposes the mechanism to realize the long-jump in the solution space. Chapter 2 is based on the papers [TSYF12, TSYF13] and Chapter 3 is based on the papers [TYSF13, TIY15].

In Chapter 4, this thesis proposes the reactive scheduling method based on LCO for the dynamic problem. The effectiveness of the proposed algorithm is verified by comparison experiments with the conventional approach based on genetic algorithm proposed by Tanimizu et al. [TSS03]. Chapter 4 is based on the papers [TYF13, TIYF14].

Finally, this thesis is summarized in Chapter 5.

1.5 Conclusion

This Chapter introduced our research background and objective. After providing the general definition of the JSP, this chapter explained the related works and the outline of this thesis. The algorithm of the conventional local clustering organization which is mainly used in this thesis is also explained in the related works.

Bibliography

[ABZ88] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Manage. Sci.*, 34(3):pp.

391–401, March 1988.

[Bak74] K.R. Baker. *Introduction to sequencing and scheduling.* Wiley, 1974.

[Bal69] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations Research*, 17(6):pp. 941–957, 1969.

[BDP96] Jacek Blazewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):pp. 1–33, 1996.

[Bie95] Christian Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, 17(2-3):pp. 87–92, 1995.

[BV98] Egon Balas and Alkis Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44(2):pp. 262–275, 1998.

[BW69] G.H. Brooks and C.R. White. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. *The Journal of Industrial Engineering*, 16(1):pp. 34–40, 1969.

[CB76] E.G. Coffman and J.L. Bruno. *Computer and job-shop scheduling theory.* A Wiley-Interscience publication. Wiley, 1976.

[CMM67] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of scheduling*. Addison-Wesley Pub. Co., 1967.

[DT93] Mauro Dell'Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):pp. 231–252, 1993.

[FMW05] Masashi Furukawa, Yusuke Matsumura, and Michiko Watanabe. Local clustering organization (LCO) solving a large scale of tsp. *Journal of Robotics and Mechatronics*, 17(5):pp. 560–567, 2005. (in Japanese).

[FMW06] Masachi Furukawa, Yusuke Matsumura, and Michiko Watanabe. Development of local clustering organization applied to job-shop scheduling problem. *Journal of the Japan Society for Precision Engineering (CD-ROM)*, 72(7):pp. 867–872, 2006. (in Japanese).

[Fre82] S. French. *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Ellis Horwood series in mathematics and its applications. E. Horwood, 1982.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[GJS76] Micheal R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:pp. 117–129, 1976.

[GT60] B. Giffler and G. L. Thompson. Algorithms for solving production-scheduling problems. *Operations Research*, 8(4):487–503, 1960.

[Koh98] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1–3):pp. 1–6, 1998.

[KS09] Yohko Konno and Keiji Suzuki. Performance of extended local clustering organization (LCO) for large scale job-shop scheduling problem (JSP). *IEEJ Transactions on Electronics Information and Systems*, 129(7):pp. 1363–1370, 2009. (in Japanese).

[LAL92] Peter J. M. van Laarhoven, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):pp. 113–125, 1992.

[NS96] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):pp. 797–813, 1996.

[NS05] Eugeniusz Nowicki and Czeslaw Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8(2):145–159, 2005.

[PI77] S. S. Panwalkar and Wafik Iskander. A survey of scheduling rules. *Operations Research*, 25(1):pp. 45–61, 1977.

[PM00] Ferdinando Pezzella and Emanuela Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2):pp. 297–310, 2000.

[RK76] A.H.G. Rinnooy Kan. *Machine scheduling problems: classification, complexity and computations*. Nijhoff, 1976.

[Tai94] E.D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6(2):pp. 108–117, 1994.

[TIY15] Yasumasa Tamura, Hiroyuki Iizuka, and Masahito Yamamoto. Extended local clustering organization with rule-based neighborhood search for job-shop scheduling problem. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2*, volume 2 of *Proceedings in Adaptation, Learning and Optimization*, pages pp. 465–477. Springer International Publishing, 2015.

[TIYF14] Yasumasa Tamura, Hiroyuki Iizuka, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization

to reactive job-shop scheduling. *Soft Computing*, pages pp. 1–9, 2014. (Published online).

[TSS03] Yoshitaka Tanimizu, Tatsuhiko Sakaguchi, and Nobuhiro Sugimura. Genetic algorithm based reactive scheduling : 1st report, modification of production schedule for delays of manufacturing processes. *Transactions of Japan Society of Mechanical Engineers C*, 69(685):pp. 2458–2463, 2003. (in Japanese).

[TSYF12] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. In *Proceedings of ASME/ISCIE 2012 International Symposium on Flexible Automation (ISFA2012)*, pages ISFA2012–7226, 2012.

[TSYF13] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. *Transactions of the Institute of Systems, Control and Information Engineers*, 26(4):pp. 121–128, 2013.

[TYF13] Yasumasa Tamura, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization to reactive job-shop scheduling. In *Proceedings of The 14th International Symposium on Advanced Intelligent Systems (ISIS2013)*, pages T1f–4, 2013.

[TYSF13] Yasumasa Tamura, Masahito Yamamoto, Ikuo Suzuki, and Masashi Furukawa. Acquisition of dispatching rules for job-shop scheduling problem by artificial neural networks using pso. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 17(5):pp. 731–738, 2013.

[WHW05] Jean-paul Watson, Adele E. Howe, and L. Darrell Whitley. Deconstructing nowicki and smutnicki's i-tsab tabu search algorithm for the job-shop scheduling problem. *Computers & Operations Research*, 33:pp. 2623–2644, 2005.

[ZF94] M. Zweben and M.S. Fox. *Intelligent Scheduling*. Morgan Kaufmann Publishers, 1994.

[ZLGR07] Chao Yong Zhang, Pei Gen Li, Zai Lin Guan, and Yun Qing Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):pp. 3229–3242, 2007.

Chapter 2

Hybrid LCO with Simulated Annealing

2.1 Introduction

Although LCO is an effective metaheuristics for the combinatorial optimization problems such as the JSP, the performance of LCO, especially for the problems which have the strong multimodality, sometimes becomes poor. On the strong multimodal problems, the search performed by LCO often is often trapped in the local optima and LCO cannot escape from there in many cases. Chapter 2 solve this problem by integrating mechanisms to escape from local optima into LCO.

In the JSP, it is known that the multimodality of the square instances is relatively stronger than that of the rectangle instances, where the rectangle

instances mean the instances in which the number of jobs are obviously larger than the number of machines. Because the size of solution space is constant when the problem size is constant, it is supposed that the distance between local optima in the strong multimodal problems is closer than that of in the weak multimodal problems. Thus, the solution trapped in the local optima can escape from there by applying small changes.

In LCO, the search of the solutions is performed greedy because 1) LCO adopts the hill climbing search to improve the solution. In addition, excepting the selection of the local, 2) all of the clustering methods in the LCO provide the deterministic neighborhood transition of the solution. From these two causes, it is supposed that LCO cannot escape from local optima.

To improve the problem 2) in the above, this chapter proposes the new clustering methods based on the Monte Carlo sampling. Also to improve the problem 1), this chapter proposes a novel hybrid algorithm based on LCO and the simulated annealing.

The rest of this chapter is organized as follows. Section 2.2 describes important features of SA. Section 2.3 proposes the new clustering methods and the hybrid algorithm based on LCO and SA. The effectiveness of the proposed algorithm is verified by the numerical experiments in Section 2.4 and 2.5. From the experimental results, the effectiveness of the proposed algorithm is discussed in Section 2.6. Finally, Section 2.7 summarizes this chapter with remarks.

This chapter is based on the papers [TSYF12, TSYF13].

2.2 Simulated Annealing

SA is a generic probabilistic metaheuristic employing local search. It consists of two principal mechanisms, called *Metropolis Monte Carlo algorithm* and *annealing schedule*, to search for the global optima. The Metropolis Monte Carlo algorithm provides the local search based on the Monte Carlo sampling and it controls the transitions of the solutions stochastically, where the probability determined by the temperature parameter T. In addition, to search for the global optima efficiently, the annealing schedule controls T on the basis of thermodynamic knowledge. To propose the hybrid algorithm based on LCO and SA, this section briefly explains these mechanism in the following parts.

2.2.1 Metropolis Monte Carlo Algorithm

The Metropolis Monte Carlo algorithm searches for the neighborhood solutions using the Monte Carlo sampling of the neighborhood solution S_n and the probabilistic trade of the current solution S_c for S_n iteratively. The Monte Carlo sampling selects a neighborhood solution from the set of neighborhood solutions for S_c at random. The current solution S_c is immediately replaced with the sampled neighborhood solution S_n in accordance with the acceptance rate P_{acc}. The acceptance rate is calculated using the Metropolis criterion shown in Equation 2.1, where ΔE denotes the energy increments in

the thermodynamic system [MRR+53].

$$P_{\mathrm{acc}}(\Delta E, T) = \begin{cases} 1.0 & \Delta E \leq 0 \\ e^{-\Delta E/T} & \Delta E > 0 \end{cases} \tag{2.1}$$

In SA for the JSP, ΔE is defined as $\Delta E = C(S_c) - C(S_n)$, where $C(S_*)$ denotes the makespan of the solution S_*. From the definition of the Metropolis criterion, the neighborhood solution which is worse than the current solution is easily accepted when T is high. In other words, when T is high, the search mechanism is almost the same as a kind of the random search, named the *multi-step random walk*. On the other hand, when T is much low, the algorithm searches for the solutions like the steepest descent method.

2.2.2 Annealing Schedule

The annealing schedule generally controls the temperature parameter T by decreasing it (from high to low) in accordance with the *cooling schedule*. The cooling schedule determines the temperature in every search step and it is one of the most important parameters for solving the problem efficiently in SA. In addition, it is theoretically proved that SA can find an optimal solution when the cooling is applied enough slowly. Equation 2.2 shows the typical cooling schedule named the *logarithmic cooling schedule*, where β denotes the constant depends on the problem and t denotes the search step.

$$T(t) = \frac{\beta}{\log(t+1)} \tag{2.2}$$

The logarithmic cooling schedule is known as one of the most effective cooling schedules because it can find the optimal solution when β is set to the ideal value. On the other hand, the logarithmic cooling schedule is usually impractical because it requires too long computation time to obtain a solution.

As a practical cooling schedule, the *linear cooling schedule* shown in Equation 2.3 is widely used in many studies, where the notation T_0 denotes the initial temperature and γ denotes the descent temperature.

$$T(t) = T_0 - \gamma t \tag{2.3}$$

The *exponential cooling schedule* (also called *geometric cooling schedule*) shown in Equation 2.4 is also used in many studies, where the notation δ denotes the decreasing rate.

$$T(t) = T_0 \delta^t \tag{2.4}$$

In SA, the annealing (cooling) is generally applied at intervals of prescribed iteration times for the search based on the Metropolis Monte Carlo algorithm. The interval, called *cooling cycle*, often affects the performance of SA, it is much difficult to determine the ideal number of iterations. One of the most typical way to determine the number of iterations is setting it to a constant. As well as other studies, this study also uses the constant interval for annealing.

2.3 Proposed Algorithms

We propose a hybrid algorithm for JSP. The proposed method is developed by combining LCO with SA.

In the previous section, it is insisted SA is a good method for generating the high accurate solution and LCO is a good method for generating the good solution with fast computation time. Therefore, by combining SA with LCO, there are some possibilities to establish a better algorithm which can generate the high accurate solution with fast computation time.

In the following, we describe the new optimization procedure and discuss the expected effects of the proposed method.

2.3.1 New Clustering Methods: REM and RIM

To increase the variation of the neighborhood solutions, this study proposes two novel clustering methods based on the Monte Carlo sampling. Some notations are given to explain the new clustering methods as follows.

$$\begin{array}{rcl} r(t) & : & \text{the clustering radius in } t^{\text{th}} \text{ step} \\ S_c & : & \text{the current solution} \\ S_n & : & \text{the neighborhood solution} \\ C(S) & : & \text{the cost of a solution } S \end{array}$$

The algorithms of the proposed clustering methods, named *REM* and *RIM*, are also described as following procedures.

Random exchange method (REM)

This method generates S_n by swapping two entities in the local of S_c at random. The procedure of SEM is described as follows;

1. Set $i = 1$.
2. Select two entities at random in the local.
3. Generate the new solution S_n by swapping selected two entities.
4. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.
5. Replace i with $i + 1$.
6. If $i \leq 2r(t)$, then go back to the step 2. Otherwise stop the procedure.

Random insertion method (RIM)

This method generates S_n by swapping two entities in the local of S_c at random. The procedure of SEM is described as follows;

1. Set $i = 1$.
2. Select two entities named x and y at random in the local.
3. Generate the new solution S_n by inserting y in front of x.
4. If $C(S_n) \leq C(S_c)$, then replace S_c with S_n.
5. Replace i with $i + 1$.
6. If $i \leq 2r(t)$, then go back to the step 2. Otherwise stop the procedure.

2.3.2 Hybrid LCO with SA

Following notations are also used in the explanation of the proposed algorithm.

$$\begin{array}{rcl} T(t) & : & \text{the temperature in } t^{\text{th}} \text{ step} \\ S_b & : & \text{the best solution} \end{array}$$

The proposed algorithm is described as following procedure.

1. Set $t = 0$.
2. Initialize S_c at random.
3. Set $S_b = S_c$.
4. Select a local, a subset of consecutive elements in S_c.
5. To create S_n, apply a clustering operation to the selected local.
6. Set $\Delta_{nc} = C(S_n) - C(S_c)$ and $\Delta_{nb} = C(S_n) - C(S_b)$.
7. If $\Delta_{nb} < 0$, replace S_b and S_c with S_n.
8. Update S_c using the temperature $T(t)$ and Δ_{nc} in accordance with the metropolis criterion.
9. Replace t with $t + 1$.
10. If terminate conditions are satisfied, stop this procedure with the solution S_b. Otherwise go back to step 4.

As well as the conventional LCO, multiple clustering methods are provided and one of the clustering methods selected stochastically is used in each step. In addition, the best solution in the search is preserved in each step because the current solution in the search can become worse due to the search mechanism inherited from SA.

2.4 Verification of the Proposed Clustering Methods

In this section, we evaluate the effectiveness of the proposed clustering methods by comparing it with the conventional clustering methods. This section only shows the experimental conditions and results. The results are discussed in the later section.

2.4.1 Experimental Conditions

To examine the effectiveness of the proposed methods, this section shows the experimental results. In the experiments, this section uses 20 well-known benchmark problems introduced by Lawrence [Law84], la16-la30 and la36-la40. To evaluate the effectiveness of each clustering method to test in different problem sizes, the benchmarks problems are classified into 4 groups in terms of the problem size. The effectiveness is evaluated using the *relative error rate RE* shown in Equation 2.5, where C_o denotes the obtained makespan

and C_{LB} denotes the lower bound of the makespan for each instance shown in the paper [JM99].

$$RE = \frac{C_o - C_{LB}}{C_{LB}} \times 100 \quad [\%] \tag{2.5}$$

In this paper, the effectiveness of the proposed clustering methods is compared with the conventional clustering methods, SEM, IEM, SYM and SIM proposed in the papers [FMW06, KS09]. All of methods are tested 100 times in each benchmark problems. The search using each clustering method is terminated when 30 seconds have elapsed from the algorithm is started.

The clustering radius $r(t)$ is determined by Equation 2.6 in each search step, where URAND(a, b) generates a uniform random number in the range $[a, b)$, n means the number of jobs and m means the number of machines.

$$r(t) = \text{URAND}\left(\frac{1}{4}, \frac{1}{3}\right) \times n \times m \tag{2.6}$$

2.4.2 Experimental Results

Figure 2.1 shows the results. In each chart, the vertical axis shows the relative error rate and the horizontal axis shows each clustering method. In the figure, REM and RIM are the proposed clustering methods in this section. From the figure, the performance of REM is almost equal to the performance shown by SEM and SYM. Also, the performance of RIM is almost equal to that of SIM.

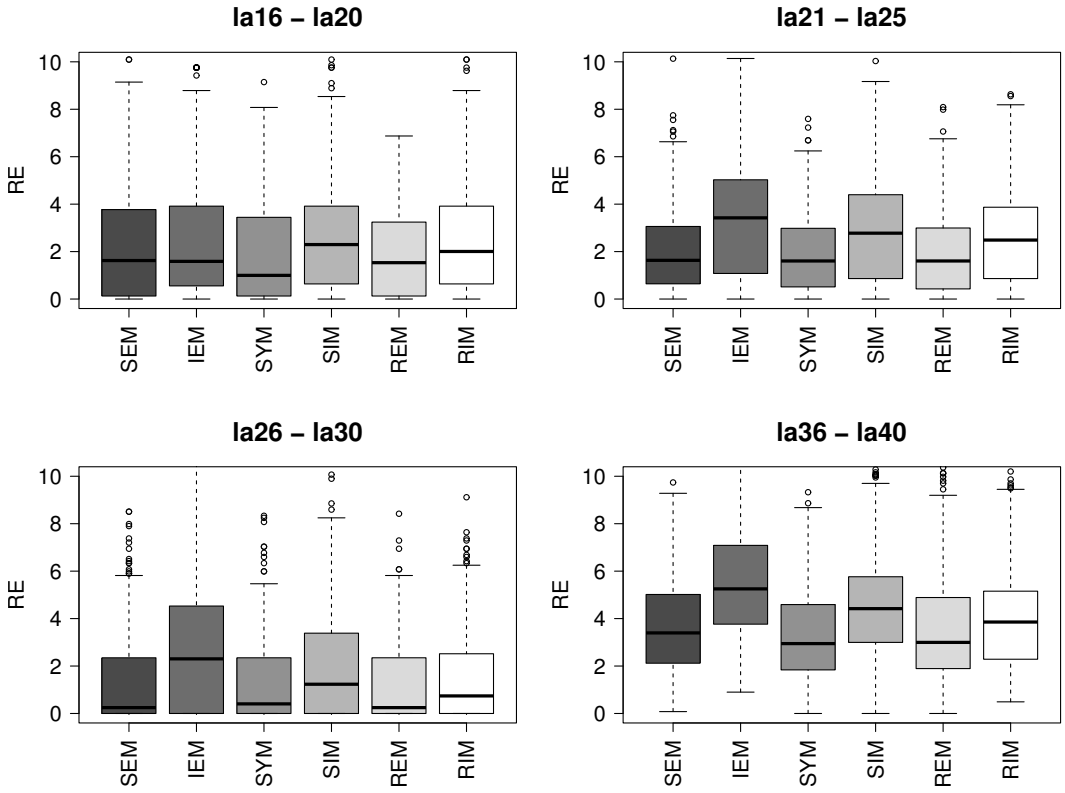


Figure 2.1: Effectiveness of the proposed clsutering methods

2.5 Verification of the Hybrid LCO with SA

To examine the effectiveness of the hybrid LCO with SA, this section also provides the experimental results. This section only shows the experimental conditions and results as well as the previous section. The discussion for the results are collected in Section 2.6.

Table 2.1: Mean computation time for SA shown in [LAL92]

group	size	computation time [sec]
la16 – la20	10×10	715.20
la21 – la25	15×10	2095.6
la26 – la30	20×10	4319.0
la31 – la35	30×10	1740.6
la36 – la40	15×15	5450.4

2.5.1 Experimental Conditions

This section uses 25 well-known benchmark instances (la16–la40) introduced by Lawrence [Law84], and 15 well-known benchmark instances (swv01–swv15) introduced by Storer et al. [SWV92]. The effectiveness of the hybrid LCO is compared with the conventional LCO, CP-based SA proposed in [LAL92] and CP-based tabu search (*TSAB*) proposed in [NS96]. Both methods, the hybrid LCO and the conventional LCO, are tested 100 times in each benchmark instance under the condition that they are terminated when 30 seconds have passed from the algorithm is started. As reference, the mean computation time for SA shown in the paper [LAL92] is described in Table 2.1.

The parameters for LCO, the clustering radius $r(t)$ and the selection probabilities for each clustering method, are determined by some preliminary experiments. The clustering radius $r(t)$ is determined as well as the previous experiment. From the results in the previous experiment, the selection probabilities are set as Equation 2.7, where (SEM), (REM) and (SYM)

respectively mean the selection probabilities of SEM, REM and SYM.

$$(\mathrm{SEM}) : (\mathrm{REM}) : (\mathrm{SYM}) = 3 : 4 : 5 \tag{2.7}$$

These parameters are also used in the hybrid LCO as well as the conventional LCO. In addition, the parameters for temperature in the hybrid LCO is determined by preliminary experiments.

2.5.2 Experimental Results: accuracy verification

To verify the effectiveness of the proposed algorithm in terms of the accuracy, the makespan of the schedule obtained by each algorithm are statistically shown in Table 2.2 and Table 2.3. On each table, *SALCO* stands for the hybrid algorithm and *LCO* stands for the conventional LCO. The results related to SA and TSAB are the reported results by each papers, [LAL92] and [NS96].

Table 2.2 shows the obtained makespan by each algorithm, where C_o^{best} and $\overline{C_o}$ means the best and mean makespan of the obtained makespan. From the table, the best and mean makespan obtained by the hybrid LCO is almost better than the one obtained by SA. Also, the best makespan obtained by the hybrid LCO is almost better than or equal to the one obtained by the conventional LCO and TSAB.

To evaluate the effectiveness of each algorithm in different instance size, the benchmark instances are divided into 8 groups in terms of the instance size. Table 2.3 shows the mean RE (MRE) of the hybrid LCO, the con-

Table 2.2: Experimental results: comparison with LCO, SA and TSAB

			SALCO		LCO		SA		TSAB
instance	$n \times m$	C_{LB}	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}
la16	10 × 10	945	945	956.88	945	971.42	956	966.2	945
la17	10 × 10	784	784	784.22	784	786.94	785	787.8	784
la18	10 × 10	848	848	849.17	848	855.43	861	861.2	848
la19	10 × 10	842	842	851.53	842	866.10	848	853.4	842
la20	10 × 10	902	902	907.32	902	922.40	902	908.4	902
la21	15 × 10	1046	1047	1061.40	1050	1077.19	1063	1067.6	1047
la22	15 × 10	927	927	934.75	927	942.02	938	944.2	927
la23	15 × 10	1032	1032	1032.06	1032	1032.26	1032	1032.0	1032
la24	15 × 10	935	938	953.73	935	961.46	952	966.6	939
la25	15 × 10	977	977	990.20	977	1000.84	992	1004.4	977
la26	20 × 10	1218	1218	1218.25	1218	1218.61	1218	1219.0	1218
la27	20 × 10	1235	1235	1253.57	1235	1261.00	1269	1273.6	1236
la28	20 × 10	1216	1216	1222.51	1216	1222.17	1224	1244.8	1216
la29	20 × 10	1152	1163	1181.26	1162	1197.44	1218	1226.4	1160
la30	20 × 10	1355	1355	1355.20	1355	1355.05	1355	1355.0	1355
la31	30 × 10	1784	1784	1784.01	1784	1784.00	1784	1784.0	1784
la32	30 × 10	1850	1850	1850.04	1850	1850.00	1850	1850.0	1850
la33	30 × 10	1719	1719	1719.00	1719	1719.00	1719	1726.6	1719
la34	30 × 10	1721	1721	1721.11	1721	1721.00	1721	1775.6	1721
la35	30 × 10	1888	1888	1888.06	1888	1888.00	1888	1890.0	1888
la36	15 × 15	1268	1274	1293.09	1269	1302.90	1293	1300.0	1268
la37	15 × 15	1397	1397	1424.42	1407	1440.96	1433	1442.4	1407
la38	15 × 15	1196	1196	1217.15	1214	1257.21	1215	1227.2	1196
la39	15 × 15	1233	1238	1248.98	1243	1277.71	1248	1258.2	1233
la40	15 × 15	1222	1224	1231.96	1225	1251.49	1234	1247.4	1229
swv01	20 × 10	1407	1429	1468.48	1445	1503.03	—	—	—
swv02	20 × 10	1475	1482	1523.20	1486	1541.83	—	—	—
swv03	20 × 10	1369	1425	1457.04	1445	1490.00	—	—	—
swv04	20 × 10	1450	1493	1520.53	1509	1567.62	—	—	—
swv05	20 × 10	1424	1443	1502.70	1497	1548.10	—	—	—
swv06	20 × 15	1591	1705	1745.55	1735	1812.69	—	—	—
swv07	20 × 15	1446	1644	1686.85	1683	1742.98	—	—	—
swv08	20 × 15	1640	1779	1838.97	1832	1903.41	—	—	—
swv09	20 × 15	1604	1690	1741.48	1739	1807.69	—	—	—
swv10	20 × 15	1631	1788	1834.00	1826	1876.20	—	—	—
swv11	50 × 10	2983	3200	3258.21	3051	3120.06	—	—	—
swv12	50 × 10	2972	3252	3315.22	3114	3189.52	—	—	—
swv13	50 × 10	3104	3298	3347.55	3194	3286.08	—	—	—
swv14	50 × 10	2968	3124	3165.38	3028	3087.48	—	—	—
swv15	50 × 10	2885	3153	3207.81	3012	3087.65	—	—	—

ventional LCO and SA for each group. The column named *Gap* shows the difference between MRE of the hybrid LCO and that of each comparison method, where ** indicates the significant difference at $p < 0.01$ and * indicates the significant difference at $p < 0.05$. From Table 2.3, the mean RE of the hybrid LCO is significantly lower than that of the conventional LCO and SA in many cases. It indicates that the hybridization obviously improves the

Table 2.3: Experimental results: mean relative error and gaps

	size	SALCO	LCO		SA	
group	$n \times m$	MRE [%]	MRE [%]	(Gap)	MRE [%]	(Gap)
la16 – la20	10×10	0.629	1.83	(**+2.21)	1.27	(** + 0.641)
la21 – la25	15×10	1.13	1.98	(**+0.845)	2.02	(** + 0.886)
la26 – la30	20×10	0.923	1.32	(**+2.21)	2.41	(** + 1.49)
la31 – la35	30×10	0.00246	0.000	(*-0.00246)	0.744	(** + 0.742)
la36 – la40	15×15	1.564	3.41	(**+1.85)	2.50	(** + 0.936)
swv01 – swv05	20×10	4.89	7.40	(**+2.51)	—	
swv06 – swv10	20×15	11.9	15.7	(**+3.75)	—	
swv11 – swv15	50×10	9.29	5.77	(**-3.53)	—	

performance of LCO and the hybrid algorithm is more effective than the SA even if SA uses the neighborhood structure based on the critical paths.

2.5.3 Experimental Results: efficiency verification

To evaluate the efficiency of the hybrid LCO, this part shows additional results. In this part, the computation time is limited 10% of the previous condition, which means the computation time is set to $3[sec]$ and this condition is called the *short term condition*.

Table 2.4 shows the results for the hybrid LCO and the conventional LCO. From the table, there are some cases where $\overline{C_o}$ of the hybrid LCO is better than that of the conventional LCO, while there are other cases where $\overline{C_o}$ of the hybrid LCO is worse than that of the conventional LCO. It indicates the hybridization affects the efficiency of searches.

Table 2.4: Experimental results: short term condition

			SALCO		LCO	
instance	$n \times m$	C_{LB}	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$
la16	10 × 10	945	945	957.58	945	974.37
la17	10 × 10	784	784	784.36	784	787.79
la18	10 × 10	848	848	849.71	848	858.68
la19	10 × 10	842	842	851.58	848	869.27
la20	10 × 10	902	902	907.32	902	925.09
la21	15 × 10	1046	1052	1067.83	1050	1082.48
la22	15 × 10	927	930	938.42	927	947.19
la23	15 × 10	1032	1032	1032.00	1032	1032.31
la24	15 × 10	935	941	962.06	941	966.63
la25	15 × 10	977	984	998.94	977	1008.69
la26	20 × 10	1218	1218	1227.75	1218	1220.61
la27	20 × 10	1235	1271	1287.42	1240	1274.67
la28	20 × 10	1216	1234	1250.83	1216	1233.63
la29	20 × 10	1152	1216	1234.02	1175	1213.75
la30	20 × 10	1355	1355	1356.90	1355	1355.12
la31	30 × 10	1784	1784	1784.00	1784	1784.00
la32	30 × 10	1850	1850	1850.00	1850	1850.00
la33	30 × 10	1719	1719	1719.00	1719	1719.00
la34	30 × 10	1721	1721	1745.81	1721	1721.00
la35	30 × 10	1888	1888	1888.00	1888	1888.00
la36	15 × 15	1268	1297	1315.95	1269	1320.04
la37	15 × 15	1397	1416	1448.26	1418	1461.26
la38	15 × 15	1196	1221	1252.40	1214	1277.14
la39	15 × 15	1233	1251	1266.17	1243	1289.14
la40	15 × 15	1222	1245	1258.93	1228	1265.79
swv01	20 × 10	1407	1490	1530.65	1459	1545.32
swv02	20 × 10	1475	1544	1577.26	1516	1571.00
swv03	20 × 10	1369	1483	1518.13	1462	1519.25
swv04	20 × 10	1450	1532	1581.89	1525	1589.74
swv05	20 × 10	1424	1518	1567.43	1512	1570.90
swv06	20 × 15	1591	1843	1901.06	1787	1868.83
swv07	20 × 15	1446	1780	1833.13	1729	1790.75
swv08	20 × 15	1640	1949	2007.89	1879	1953.60
swv09	20 × 15	1604	1805	1889.25	1753	1851.35
swv10	20 × 15	1631	1900	1974.35	1851	1924.67
swv11	50 × 10	2983	3574	3658.26	3203	3287.24
swv12	50 × 10	2972	3629	3719.03	3279	3352.18
swv13	50 × 10	3104	3606	3733.28	3317	3414.84
swv14	50 × 10	2968	3502	3565.35	3134	3239.12
swv15	50 × 10	2885	3511	3613.79	3149	3250.31

2.6 Discussion

From the results on the verification of the proposed clustering methods, the deterministic clustering used in the conventional LCO does not affect the performance so much. In other words, it is proved that the conventional clustering methods can search for enough neighborhood solutions. On the other hand, the performance of the proposed clustering methods are not worse than

the other methods based on the same operator to generate a neighborhood solution. Thus, the performance of the neighborhood search depends on the operator to generate a neighborhood solution, and for the permutation with repetition, the operator which generates neighborhood solution by swapping two entities is the most effective operator in this research.

From the results on the verification of the hybrid algorithm, it is proved that the hybridization obviously improves the performance of LCO. In addition, surprisingly, the performance of the hybrid algorithm is significantly higher that that of SA and the hybrid algorithm is capable of obtaining as good solution as TSAB which is one of the most effective algorithms employing local search for the JSP. On the other hand, in short term condition, the makespan obtained by the hybrid LCO for la26-la35 and swv06-swv15 shown in Table 2.4 is averagely worse than the conventional LCO. From the view point of the problem size, the number of jobs is obviously larger than the number of jobs in these instances. Thus, it is expected that the hybrid LCO searches the solutions inefficiently because those instances has weak multimodality. This result proves that the effective mechanism to escape from local optima can be different depending on the multimodality, and it is assumed that the suitable mechanism to escape from local optima makes searches efficient. Finally, from the experimental results in this chapter, it is also proved that the mechanism to escape from local optima based on SA is suitable for the strong multimodal problems.

2.7 Conclusion

This chapter is summarized as follows;

- This chapter proposes new clustering methods based on the Monte Carlo sampling.
- The effectiveness of the proposed clustering methods is verified in the numerical experiments.
- This chapter also proposes a hybrid algorithm based on LCO and SA for the JSP.
- The effectiveness of the proposed hybrid algorithm is also verified in the numerical experiments.

This chapter proves some important remarks as follows;

- The performance of the local search depends on the operator to generate a neighborhood solution.
- The proposed hybrid algorithm is more effective than that uses critical paths and it suggests that the local search without critical paths can be more effective than the other algorithms with critical paths.
- The effective mechanism to escape from the local optima can be different depending on the multimodality.

Bibliography

[FMW06] Masachi Furukawa, Yusuke Matsumura, and Michiko Watanabe. Development of local clustering organization applied to job-shop scheduling problem. *Journal of the Japan Society for Precision Engineering (CD-ROM)*, 72(7):pp. 867–872, 2006. (in Japanese).

[JM99] A.S. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):pp. 390–434, 1999.

[KS09] Yohko Konno and Keiji Suzuki. Performance of extended local clustering organization (LCO) for large scale job-shop scheduling problem (JSP). *IEEJ Transactions on Electronics Information and Systems*, 129(7):pp. 1363–1370, 2009. (in Japanese).

[LAL92] Peter J. M. van Laarhoven, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):pp. 113–125, 1992.

[Law84] S. Lawrence. Resource constrained project scheduling: An experimental investigation of heuristics scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1984.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state

calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):pp. 1087–1092, 1953.

[NS96] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):pp. 797–813, 1996.

[SWV92] Robert H. Storer, S. David Wu, and Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495–1509, 1992.

[TSYF12] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. In *Proceedings of ASME/ISCIE 2012 International Symposium on Flexible Automation (ISFA2012)*, pages ISFA2012–7226, 2012.

[TSYF13] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. *Transactions of the Institute of Systems, Control and Information Engineers*, 26(4):pp. 121–128, 2013.

Chapter 3

Advanced Metaheuristics using Heuristic Perturbation

3.1 Introduction

In the previous chapter, this thesis shows that controlling the transition of the solutions using the Metropolis criterion is enough effective to escape from local optima especially for the square instances. In Chapter 3, this thesis discusses the effective mechanism to escape from local optima especially for the rectangle instances.

In general, it is known that the multimodality of the rectangle instances is relatively weak. Therefore, it is supposed that the fitness landscape of the rectangle instances has several large valleys and the distance between each local optima is longer than the square instances.

The rest of this chapter is organized as follows. In Section 3.2, this thesis proposes a new perturbation method to change the schedule considerably. Section 3.2 also proposes a novel neighborhood search algorithm based on the priority rules to make the effectiveness higher than the new perturbation method. To verify the effectiveness of the proposed perturbations, Section 3.3 shows the results of the numerical experiments. After the discussion in Section 3.4, this chapter is summarized in Section 3.5.

This chapter is based on [TYSF13, TIY15].

3.2 Extended LCO with Perturbation

This thesis integrates a novel perturbation mechanism into LCO. Firstly, this section proposes the perturbation method based on the partial rearrangements of job sequences on the solution representation. Then, this section also proposes the perturbation using priority rules. Finally, by the integration of the proposed perturbation and LCO, this section proposes the extended LCO.

3.2.1 Block-based Perturbation

To apply the effective perturbation, this study provides the *perturbation blocks*. The perturbation blocks are defined as the subsequences of the solution representation which consist of the consecutive jobs processed on the same machines. By the evaluation of the solution representation, each job

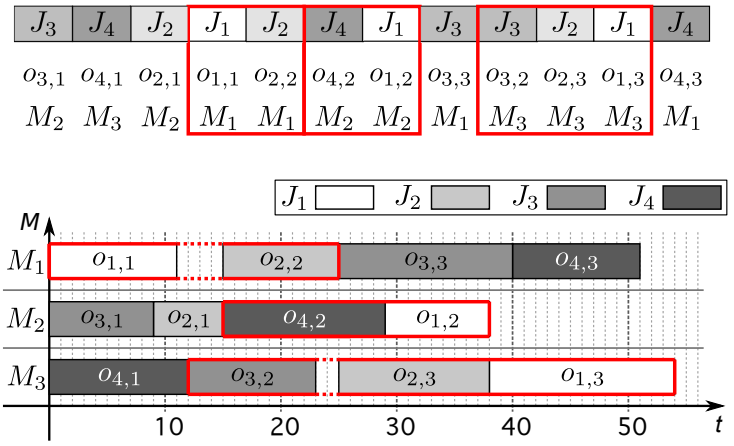


Figure 3.1: Definition of the perturbation blocks

in the permutation with repetition is associated with the machine on which the job is processed. The association between jobs and machines defines the sequence of consecutive jobs which are processed on the same machine as the perturbation blocks. For example, Figure 3.1 shows the perturbation blocks on a solution representation. There are three perturbation blocks, $\{o_{1,1}, o_{2,2}\}$, $\{o_{4,2}, o_{1,2}\}$ and $\{o_{3,2}, o_{2,3}, o_{1,3}\}$, marked with the red lines.

This study defines the perturbation as the rearrangements of jobs only in the blocks, and the rearrangements are applied to each block independently and simultaneously. It is important that the rearrangement of job sequences in the rearrangeable blocks certainly changes the schedule. From the definition of the perturbation, the perturbation method behaves like a neighborhood search algorithm which is named *large-scale neighborhood search*. However, because there are usually multiple perturbation blocks in a solution,

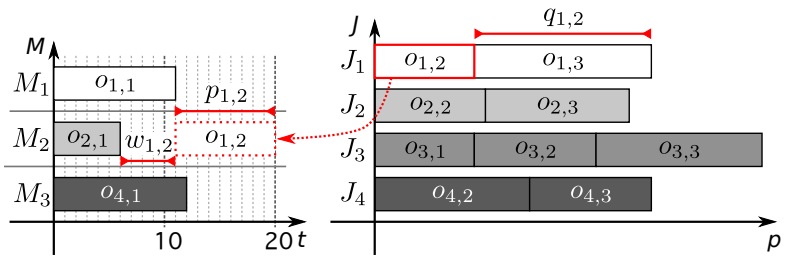


Figure 3.2: An example of rule-based schedule construction

the number of neighborhood solutions generated from a solution sometimes increases exponentially. Therefore, it is impractical to enumerate all of the neighborhood solutions. This study defines that the perturbation method selects the limited number of neighborhood solution at random per step. The perturbation method described in this part is named the *block-based perturbation* and the number of selected neighborhood solution is usually set to one.

3.2.2 Perturbation with Priority Rules

For the effective perturbation, it is important to select a promising neighborhood solution. To realize such a selection, this study applies the priority rules. The proposed method in this part is named the *perturbation with priority rules* and it uses the priority rules to select the limited number of neighborhood solutions and to acquire the effective neighborhood solutions efficiently.

To select the limited number of promising neighborhood solutions, the

proposed algorithm uses the *linear composite rule*. The linear composite rule consists of several simple priority rules, such as the *shortest processing time* (SPT) and the *least work remains* (LWKR), and it gives the priority $v(o_{i,k})$ to each operation $o_{j,k}$. This study uses following three notations to describe the composite priority rules.

$p_{i,k}$: the processing time of the operation $o_{i,k}$
$q_{i,k}$: the total processing time of the following operations of $o_{i,k}$
$w_{i,k}$: the waiting time of $o_{i,k}$ for the machines

From the definition, $q_{i,k}$ is calculated as Equation 3.1, where the notation m denotes the number of machines.

$$q_{i,k} = \sum_{l=k+1}^{m} p_{i,l} \tag{3.1}$$

The example of those notations for the operation $o_{1,2}$ is shown in Figure 3.2. It is commonly known that $p_{i,k}$ is used to describe the simple priority rule *SPT*, $q_{i,k}$ is used to describe the simple priority rule *LWKR* and $w_{i,k}$ is used to describe a simple priority rule named the *first come first serve (FCFS)*. With these simple rules, the linear composite rule is defined as the linear combination shown in Equation 3.2, where the notations α_1, α_2 and α_3 correspond to the weights.

$$v(o_{i,k}) = \alpha_1 \frac{p_{i,k}}{\max p_{i,k}} + \alpha_2 \frac{q_{i,r}}{\max q_{i,k}} + \alpha_3 \frac{w_{i,k}}{\max w_{i,k}} \tag{3.2}$$

It is verified that the linear composite rule can construct relatively good schedule, when the weights $(\alpha_1, \alpha_2, \alpha_3)$ are optimized by the unsupervised

learning. The detail of the learning algorithm with the *particle swarm optimization* and the effectiveness of the composite priority rules are discussed in the paper [TYSF13].

The perturbation with priority rules generates neighborhood solutions using the liner composite rule. The sequence of jobs in each perturbation block is sorted by the priority value of each job (operation). The priority rule used in this study is dynamically determined by the real coded vector $(\alpha_1, \alpha_2, \alpha_3)$.

A set of priority rules is created by generating individual vectors $(\alpha_1, \alpha_2, \alpha_3)$. The number of priority rules is a parameter of the proposed method and it corresponds to the number of neighborhood solutions searched per step. In addition, because the effective rules can be different in each instance and the best rule is generally unpredictable, the rules are optimized along with the searches of the solutions (schedules). In this study, the optimization of the rules is performed by genetic algorithm (GA). The fitness of each rule is evaluated by the makespan of the solution generated by the rule. From this mechanism, some effective rules which generate effective neighborhood solutions are adaptively obtained in searches, and the optimization of the schedule is progressed using the effective rules.

3.2.3 Integration into LCO

This study proposes two solution algorithms. The one is named the extended LCO using random perturbation and the other is named the extended LCO

using perturbation with priority rules. The rest of this part explains each algorithm.

Extended LCO using Block-based Perturbation

The following procedure briefly shows the algorithm.

1. Generate an initial solution S_c at random.
2. Set $s \leftarrow 0$
3. Select a local on S_c and a clustering method stochastically.
4. Apply the selected clustering method to the selected local on S_c.
5. If the makespan is improved, set $s \leftarrow 0$. Otherwise set $s \leftarrow s + 1$.
6. If $s > s_{\max}$, apply the random perturbation to S_c.
   Then, set $s \leftarrow 0$ and $s_{\max} \leftarrow \gamma \times s_{\max}$.
7. If termination conditions of the algorithm are satisfied, stop the procedure. Otherwise, go back to step 3.

In the procedure, the steps 3-4 correspond to the searches performed by the conventional LCO and the steps 5-6 correspond to the random perturbation. The application cycle of the random perturbation is controlled by the stagnation count. The maximum stagnation count is also controlled by γ. In addition, because the current solution can become worse in the step 6, the best solution found in searches is memorized.

Extended LCO using Perturbation with Priority Rules

The following procedure briefly shows the algorithm.

1. Generate an initial solution S_c at random.
2. Generate initial rules at random and evaluate each rule.
3. Generate a uniform random value u.
4. If $u < 1 - \varepsilon$, go to step 5. Otherwise, go to step 7.
5. Select a local on S_c and a clustering method stochastically.
6. Apply the selected clustering method to the selected local on S_c. Then, go to step 10.
7. Update the rules.
8. Generate neighborhood solutions from S_c using each rule and replace the best neighborhood solution with S_c.
9. Evaluate each rule.
10. If termination conditions of the algorithm are satisfied, stop the procedure. Otherwise, go back to step 3.

In the procedure, the steps 5-6 correspond to the searches performed by the conventional LCO and the steps 7-9 correspond to the perturbation with priority rules. The step 7 and step 9 correspond to the optimization processes for the rules in the procedure. The optimization of the rules is performed

by the genetic algorithm (GA). In this algorithm, the best solution found in searches is also memorized.

3.3 Numerical Experiments

To discuss the effectiveness of the proposed method, this section provides the experimental results. In the experiments, this paper uses 25 well-known benchmark instances (la16-la40) introduced by Lawrence [Law84]. The effectiveness of the extended LCO is compared with the conventional LCO, CP-based SA proposed in [LAL92] and CP-based tabu search ($TSAB$) proposed in [NS96]. The conventional LCO and the extended LCO are terminated when 30 seconds have passed from the algorithm is started.

The parameters for LCO, the clustering radius $r(t)$ and the selection probabilities for each clustering method, are determined by some preliminary experiments. The clustering radius $r(t)$ is determined by Equation 3.3 in each search step, where URAND(a, b) generates a uniform random number in the range $[a, b)$, n means the number of jobs and m means the number of machines.

$$r(t) = \text{URAND}\left(\frac{1}{4}, \frac{1}{3}\right) \times n \times m \tag{3.3}$$

The effectiveness of each individual clustering method, SEM, SYM and REM is also verified in the preliminary experiments. Considering the effectiveness of LCO, the selection probabilities should generally be set to (SEM) $\geq$ (REM) $\gg$ (SYM), where (SEM), (REM) and (SYM) respectively mean the

selection probabilities of SEM, REM and SYM. In this study, the selection probabilities are set as Equation 3.4.

$$(\text{SEM}) : (\text{REM}) : (\text{SYM}) = 3 : 4 : 5 \tag{3.4}$$

These parameters are also used in the extended LCO as well as the conventional LCO. In addition, the extended LCO using perturbation with priority rules generates 10 neighborhood solutions using 10 kinds of the priority rules in the searching processes to apply the proposed neighborhood search.

3.3.1 The Effectiveness of the Block-based Perturbation

Firstly, we evaluate the block-based perturbation by comparing it with the conventional LCO, CP-based SA and TSAB. Table 3.1 shows the best and mean makespan of obtained schedule for each instance, where *ExLCO* means the extended LCO and *LCO* means the conventional LCO. C_o^{best} and $\overline{C_o}$ for the extended LCO and the conventional LCO means the best and mean makespan over 100 trials. On the other hand, C_o^{best} and $\overline{C_o}$ for SA are reported in the paper [LAL92], and C_o^{best} for TSAB is also reported in the paper [NS96]. In addition, Table 3.2 shows the mean relative error rate (MRE) for each instance group which divided by the problem size. The relative error rate RE is calculated by Equation 3.5, where C_o means the obtained makespan and C_{LB} means the known lower bound of the makespan for each instance.

$$RE = \frac{C_o - C_{LB}}{C_{LB}} \times 100[\%] \tag{3.5}$$

Table 3.1: Experimental results: comparison with LCO, SA and TSAB

			ExLCO		LCO		SA		TSAB
instance	$n \times m$	C_{LB}	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}
la16	10 × 10	945	945	963.71	945	971.42	956	966.2	945
la17	10 × 10	784	784	785.05	784	786.94	785	787.8	784
la18	10 × 10	848	848	851.86	848	855.43	861	861.2	848
la19	10 × 10	842	842	853.72	842	866.10	848	853.4	842
la20	10 × 10	902	902	910.72	902	922.40	902	908.4	902
la21	15 × 10	1046	1047	1063.28	1050	1077.19	1063	1067.6	1047
la22	15 × 10	927	927	934.47	927	942.02	938	944.2	927
la23	15 × 10	1032	1032	1032.00	1032	1032.26	1032	1032.0	1032
la24	15 × 10	935	938	948.67	935	961.46	952	966.6	939
la25	15 × 10	977	977	989.06	977	1000.84	992	1004.4	977
la26	20 × 10	1218	1218	1218.17	1218	1218.61	1218	1219.0	1218
la27	20 × 10	1235	1235	1253.72	1235	1261.00	1269	1273.6	1236
la28	20 × 10	1216	1216	1218.11	1216	1222.17	1224	1244.8	1216
la29	20 × 10	1152	1153	1183.94	1162	1197.44	1218	1226.4	1160
la30	20 × 10	1355	1355	1355.00	1355	1355.05	1355	1355.0	1355
la31	30 × 10	1784	1784	1784.00	1784	1784.00	1784	1784.0	1784
la32	30 × 10	1850	1850	1850.00	1850	1850.00	1850	1850.0	1850
la33	30 × 10	1719	1719	1719.00	1719	1719.00	1719	1726.6	1719
la34	30 × 10	1721	1721	1721.00	1721	1721.00	1721	1775.6	1721
la35	30 × 10	1888	1888	1888.00	1888	1888.00	1888	1890.0	1888
la36	15 × 15	1268	1268	1292.82	1269	1302.90	1293	1300.0	1268
la37	15 × 15	1397	1397	1428.35	1407	1440.96	1433	1442.4	1407
la38	15 × 15	1196	1202	1241.13	1214	1257.21	1215	1227.2	1196
la39	15 × 15	1233	1233	1262.06	1243	1277.71	1248	1258.2	1233
la40	15 × 15	1222	1224	1241.83	1225	1251.49	1234	1247.4	1229

Table 3.2: Experimental results: mean relative error and gaps

	size	ExLCO	LCO		SA	
group	$n \times m$	MRE [%]	MRE [%]	(Gap)	MRE [%]	(Gap)
la16 – la20	10 × 10	0.986	1.83	(**+0.849)	1.27	(** + 0.285)
la21 – la25	15 × 10	1.03	1.98	(**+0.948)	2.02	(** + 0.989)
la26 – la30	20 × 10	0.895	1.32	(**+0.427)	2.41	(** + 1.38)
la31 – la35	30 × 10	0.000	0.000	(± 0.000)	0.744	(** + 0.744)
la36 – la40	15 × 15	2.39	3.41	(**+1.02)	2.50	(+0.109)

Table 3.2 also shown the difference of MRE between the extended LCO and the comparison algorithms, and the gaps which have significant difference

at $p < 0.01$ are marked by **.

From Table 3.1, C_o^{best} and $\overline{C_o}$ of the extended LCO is better than or almost equal to that of the conventional LCO and SA. This trend is also shown by Table 3.2 statistically. Additionally, Table 3.1 shows that C_o^{best} of the extended LCO is better than that of TSAB in some cases. These results prove the effectiveness of the extended LCO using block-based perturbation.

3.3.2 The Effectiveness of the Perturbation with Priority Rules

Secondly, we evaluate the perturbation with priority rules by (ExLCO using PPR) comparing it with the block-based perturbation (ExLCO using BBP), CP-based SA [LAL92] and TSAB [NS96] as well as the above part.

Table 3.3 shows C_o^{best} and $\overline{C_o}$ of each algorithm, and Table 3.4 shows the mean relative error rate MRE of the extended LCO using PPR, the extended LCO using BBP and SA reported by [LAL92]. In Table 3.4, ** indicates the significant difference at $p < 0.01$ as well as previous results, while † indicates the significant difference at $p < 0.10$.

From these tables, it is proved that the extended LCO using perturbation with priority rules (ExLCO using PPR) can averagely obtain better solution than the extended LCO using block-based perturbation (ExLCO using BBP). Moreover, from Table 3.4, the MRE of ExLCO using PPR is much lower than that of ExLCO using BBP especially in the instances la16-la20 and la36-la40.

Table 3.3: Experimental results: comparison with BBP, SA and TSAB

			ExLCO using PPR		ExLCO using BBP		SA		TSAB
instance	$n \times m$	C_{LB}	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}
la16	10 × 10	945	945	953.94	945	963.71	956	966.2	945
la17	10 × 10	784	784	784.18	784	785.05	785	787.8	784
la18	10 × 10	848	848	848.65	848	851.86	861	861.2	848
la19	10 × 10	842	842	846.89	842	853.72	848	853.4	842
la20	10 × 10	902	902	904.19	902	910.72	902	908.4	902
la21	15 × 10	1046	1047	1058.66	1047	1063.28	1063	1067.6	1047
la22	15 × 10	927	927	932.29	927	934.47	938	944.2	927
la23	15 × 10	1032	1032	1032.00	1032	1032.00	1032	1032.0	1032
la24	15 × 10	935	935	944.05	938	948.67	952	966.6	939
la25	15 × 10	977	977	983.24	977	989.06	992	1004.4	977
la26	20 × 10	1218	1218	1218.00	1218	1218.17	1218	1219.0	1218
la27	20 × 10	1235	1235	1251.22	1235	1253.72	1269	1273.6	1236
la28	20 × 10	1216	1216	1217.61	1216	1218.11	1224	1244.8	1216
la29	20 × 10	1152	1163	1179.31	1153	1183.94	1218	1226.4	1160
la30	20 × 10	1355	1355	1355.00	1355	1355.00	1355	1355.0	1355
la31	30 × 10	1784	1784	1784.00	1784	1784.00	1784	1784.0	1784
la32	30 × 10	1850	1850	1850.00	1850	1850.00	1850	1850.0	1850
la33	30 × 10	1719	1719	1719.00	1719	1719.00	1719	1726.6	1719
la34	30 × 10	1721	1721	1721.00	1721	1721.00	1721	1775.6	1721
la35	30 × 10	1888	1888	1888.00	1888	1888.00	1888	1890.0	1888
la36	15 × 15	1268	1268	1291.16	1268	1292.82	1293	1300.0	1268
la37	15 × 15	1397	1397	1420.83	1397	1428.35	1433	1442.4	1407
la38	15 × 15	1196	1201	1228.95	1202	1241.13	1215	1227.2	1196
la39	15 × 15	1233	1233	1251.77	1233	1262.06	1248	1258.2	1233
la40	15 × 15	1222	1224	1234.69	1224	1241.83	1234	1247.4	1229

Table 3.4: Experimental results: mean relative error and gaps

	size	ExLCO using PPR	ExLCO using BBP		SA	
group	$n \times m$	MRE [%]	MRE [%]	(Gap)	MRE [%]	(Gap)
la16 – la20	10 × 10	0.374	0.986	(**+0.612)	1.27	(** + 0.896)
la21 – la25	15 × 10	0.678	1.03	(**+0.354)	2.02	(** + 1.34)
la26 – la30	20 × 10	0.763	0.895	(†+0.132)	2.41	(** + 1.65)
la31 – la35	30 × 10	0.000	0.000	(± 0.000)	0.744	(** + 0.744)
la36 – la40	15 × 15	1.77	2.39	(**+0.621)	2.50	(** + 0.730)

Table 3.5: Experimental results: short term condition

			ExLCO using PPR		ExLCO using BBP		SALCO		LCO	
instance	$n \times m$	C_{LB}	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$	C_o^{best}	$\overline{C_o}$
la16	10 × 10	945	945	966.07	945	969.43	945	957.58	945	974.37
la17	10 × 10	784	784	784.90	784	786.62	784	784.36	784	787.79
la18	10 × 10	848	848	852.01	848	856.27	848	849.71	848	858.68
la19	10 × 10	842	842	853.00	842	858.26	842	851.58	848	869.27
la20	10 × 10	902	902	910.61	902	914.77	902	907.32	902	925.09
la21	15 × 10	1046	1047	1068.82	1047	1071.34	1052	1067.83	1050	1082.48
la22	15 × 10	927	927	937.54	927	938.65	930	938.42	927	947.19
la23	15 × 10	1032	1032	1032.00	1032	1032.00	1032	1032.00	1032	1032.31
la24	15 × 10	935	936	958.52	940	957.08	941	962.06	941	966.63
la25	15 × 10	977	977	996.63	981	1000.47	984	998.94	977	1008.69
la26	20 × 10	1218	1218	1219.09	1218	1219.76	1218	1227.75	1218	1220.61
la27	20 × 10	1235	1243	1267.87	1235	1266.82	1271	1287.42	1240	1274.67
la28	20 × 10	1216	1216	1228.22	1216	1226.88	1234	1250.83	1216	1233.63
la29	20 × 10	1152	1180	1206.53	1173	1207.34	1216	1234.02	1175	1213.75
la30	20 × 10	1355	1355	1355.13	1355	1355.00	1355	1356.90	1355	1355.12
la31	30 × 10	1784	1784	1784.00	1784	1784.00	1784	1784.00	1784	1784.00
la32	30 × 10	1850	1850	1850.00	1850	1850.00	1850	1850.00	1850	1850.00
la33	30 × 10	1719	1719	1719.00	1719	1719.00	1719	1719.00	1719	1719.00
la34	30 × 10	1721	1721	1721.00	1721	1721.00	1721	1745.81	1721	1721.00
la35	30 × 10	1888	1888	1888.00	1888	1888.00	1888	1888.00	1888	1888.00
la36	15 × 15	1268	1278	1312.18	1275	1312.13	1297	1315.95	1269	1320.04
la37	15 × 15	1397	1407	1450.62	1412	1457.04	1416	1448.26	1418	1461.26
la38	15 × 15	1196	1209	1263.09	1223	1267.15	1221	1252.40	1214	1277.14
la39	15 × 15	1233	1249	1279.91	1246	1281.14	1251	1266.17	1243	1289.14
la40	15 × 15	1222	1229	1260.17	1228	1261.20	1245	1258.93	1228	1265.79

Additionally, Table 3.5 shows the results in the shot term condition as well as Part 2.5.3. The short term condition means that the computation time is limited to 10% of the original condition. It also means the results shown in Table 3.5 are obtained by each algorithm under the condition that the computation time is set to 3 [*sec*] because the original computation time is set to 30 [*sec*].

From Table 3.5, though C_o^{best} of each algorithm is almost the same, $\overline{C_o}$ of ExLCO using PPR is lower than that of the conventional LCO in many

cases. Also, $\overline{C_o}$ of ExLCO using PPR is lower than that of ExLCO using BBP and SALCO, the proposed algorithm in Chapter 2, in some cases. The remarkable results are that $\overline{C_o}$ of ExLCO using PPR or BBP is lower than that of SALCO in the instances la21-la30, while $\overline{C_o}$ of ExLCO using PPR or BBP is higher than that of SALCO in the instances la16-la20 and la36-la40. It is assumed that this trend indicates the association between multimodality of each instance and effective mechanism to escape from local optima.

3.4 Discussion

Firstly, we discuss the effectiveness of the extended LCO using block-based perturbation (ExLCO using BBP). From the experimental results, it is proved that the effectiveness of ExLCO using BBP is averagely higher than the conventional LCO and SA. It also proves that the block-based perturbation improves the performance of LCO. What remarkable is that the effectiveness of ExLCO using BBP is much higher than SA especially in the instances la21-la35. In the instances la21-la35, the number of jobs is larger than the number of machines, and it means the instances are rectangle instances which have relatively weak multimodality. In addition to these results, the effectiveness of the conventional LCO is also much higher than SA especially in the instances la26-la35. These results suggest that ExLCO using BBP inherits characteristics of the conventional LCO and the block-based perturbation makes the search performance higher especially in the instances which have

weak multimodality.

Secondly, we discuss the effectiveness of the extended LCO using perturbation with priority rules (ExLCO using PPR). From the experimental results, it is proved that the effectiveness of ExLCO using PPR is higher than ExLCO using BBP and SA. It proves that the perturbation with priority rules can effectively search for better solutions than the block-based perturbation. The difference between the block-based perturbation and the perturbation with priority rules is the existence of the limitation of neighborhood solutions except for the limitation comes from the perturbation blocks. Therefore, it is suggested that the rule-based limitation of the neighborhood solutions leads effective searches.

Finally, we discuss the efficiency of searches performed by ExLCO. From the results under the short term condition, $\overline{C_o}$ of ExLCO is averagely better than that of SALCO and the conventional LCO especially in the instances la21-la30, while $\overline{C_o}$ of ExLCO is averagely worse than that of SALCO especially in the instances la16-la20 and la36-la40. These results proves that the efficient mechanism to escape from local optima depends on the strength of multimodality.

3.5 Conclusion

This chapter proposes an improved LCO integrated with a novel large-scale neighborhood search method. The large-scale neighborhood search method

has some significant characteristics as follows:

- This chapter defines a novel neighborhood structure to change the current solution drastically and without fail.
- This chapter also proposes a novel method to generate neighborhood solutions with priority rules.
- Effective neighborhood solutions are generated at random or using some dynamic priority rules optimized along with searches of the schedule.
- The number of neighborhood solutions are limited by the number of rules not to deteriorate the efficiency of searches.

The effectiveness of the improved LCO is verified by some numerical experiments. The experimental results are summarized as follows:

- The proposed mechanisms to escape from local optima are effective especially for the instances which have relatively weak multimodality.
- This chapter proves that the specific rules to select effective neighborhoods can be extracted from search processes.
- It is also proved that the restriction of the neighborhood solution with the extracted rules provides effective search mechanism to escape from local optima.
- The comparison of the proposed algorithm in this chapter with the hybrid LCO with SA proposed in Chapter 2 suggests that the effective

mechanism to escape from local optima depends on the strength of multimodality.

Bibliography

[LAL92] Peter J. M. van Laarhoven, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):pp. 113–125, 1992.

[Law84] S. Lawrence. Resource constrained project scheduling: An experimental investigation of heuristics scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1984.

[NS96] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):pp. 797–813, 1996.

[TIY15] Yasumasa Tamura, Hiroyuki Iizuka, and Masahito Yamamoto. Extended local clustering organization with rule-based neighborhood search for job-shop scheduling problem. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2*, volume 2 of *Proceedings in Adaptation, Learning and Optimization*, pages pp. 465–477. Springer International Publishing, 2015.

[TYSF13] Yasumasa Tamura, Masahito Yamamoto, Ikuo Suzuki, and Masashi Furukawa. Acquisition of dispatching rules for job-shop scheduling problem by artificial neural networks using pso. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 17(5):pp. 731–738, 2013.

Chapter 4

Ractive Scheduling using LCO

4.1 Introduction

It is normally assumed that the given information or states in the JSP, such as the number of jobs, the number of machines, the processing time, the technological sequences and so on, are immutable. The scheduling problems on the basis of such assumption are generally called the *static scheduling problem*. The well-studied algorithms for the JSP usually focus on the static scheduling to determine a schedule with short computation time before the production is begun, which is called pre-scheduling. On the other hand, especially in the practical manufacturing systems, there are many unpredictable situations such as machine breakdowns, process delays, the additional tasks and so on. These unpredictable situations make the predetermined schedule change dynamically, and as a result of such changes, the efficiency of the

production schedule is often deteriorated fatally. The scheduling problems which assume such a dynamic environment are called the *dynamic scheduling problem*. To keep efficiency of the production schedule, not only the algorithms for the static scheduling, the algorithms for the dynamic scheduling are also important in the manufacturing systems.

Reactive scheduling (RS) is a methodology to modify the predetermined schedule depending on the occurrence of the unpredictable changes. Compared to the algorithms for the static scheduling, the modification process, called *RS process* in this thesis, requires a fast optimization algorithm to improve the schedule without suspending the production line. The classical approaches for the RS are generally based on the priority rules in terms of the computation time. The RS process using priority rules can immediately modify the schedule indeed, however the quality of the modified schedule is often poor. Tanimizu et al. focused on the metaheuristics in terms of the fast computation and they proposed the RS process using genetic algorithm (GA) as a preceding study on the application of the metaheuristics to the RS [TSS03]. The effectiveness of their method is verified by some comparative experiments using the classical RS process using priority rules.

As a metaheuristic method for the static JSP, LCO is more effective algorithm than GA in terms of the accuracy and the computational time. It means LCO has a possibility to improve the performance of the RS process. This study proposes a novel method for the RS process using LCO. To apply LCO to the RS process, this study provides LCO with the mechanism to deal

with the solution representation which dynamically changes. In addition, this paper verifies the effectiveness of the proposed method by numerical experiments.

The rest of this chapter is organized as follows. Section 4.2 describes the dynamic scheduling problem which this study focuses on. Section 4.3 proposes an optimization algorithm based on LCO for the RS. In order to analyze the effectiveness of the proposed algorithm, this paper shows several results of the numerical experiments in Section 4.4. Section 4.5 discusses the effectiveness of the proposed algorithm. Finally, Section 4.6 summarizes this chapter.

This chapter is based on the papers [TYF13, TIYF14].

4.2 Reactive Job-shop Scheduling

Because the dynamic scheduling problem in this chapter is fundamentally based on the JSP, the problem this chapter deals with is called the *reactive job-shop scheduling problem*. The unpredictable changes of the predetermined schedule in the manufacturing systems are typically caused by the processing delay or the additional tasks. Although, the occurrence of the additional tasks largely affects the efficiency of the predetermined schedule, the RS process for the additional tasks can be operated by the same procedure for the processing delay because there is no difference except the determination of the initial schedule for the RS process. To focus on the performance of

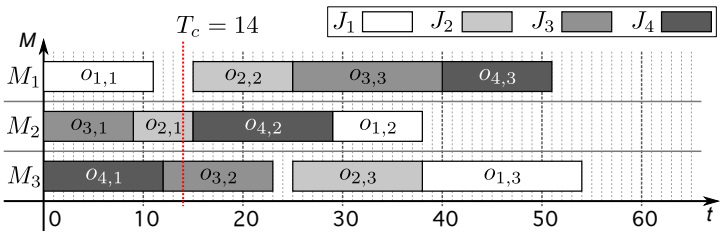
LCO to modify the predetermined schedule, this chapter particularly deals with the unpredictable changes caused by the processing delay.

4.2.1 Triggers for the RS Process

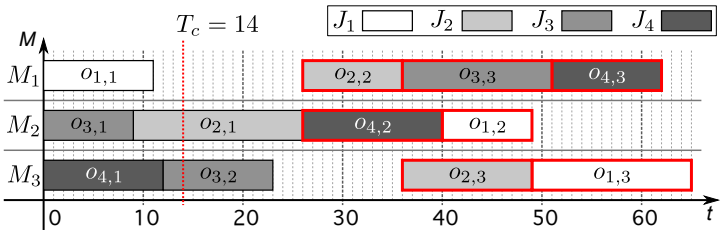
The unpredictable change for the predetermined schedule is caused by the processing delay in this study. The initial condition and the terminal condition of the RS process are defined as follows.

Initial condition

The RS process starts when the processing time of some operations are prolonged in the production activity and the efficiency of the predetermined schedule becomes worse under the influence of the changes. In order to define the start condition of the RS process, this study provides a notation $T_c > 0$ which means the time when the changes occur. For instance, Figure 4.1(a) and 4.1(b) show the example of the change at time T_c. It becomes obvious that the processing time of $o_{2,1}$ shown in Figure 4.1(a) is prolonged at T_c due to certain troubles, which results in the prolonged schedule shown in Figure 4.1(b). Depending on the change, the makespan of the predetermined schedule is also prolonged. If the new makespan is longer than the original makespan, the RS process is started.



(a) Before changing



(b) After changing

Figure 4.1: Change of schedule at the time T_c

Terminal condition

As the result of applying the RS process to the changed schedule, if the makespan of the updated schedule becomes equal to the original makespan, the RS process is terminated. Also, if there is no operation to modify the sequence on each machine, the RS process is terminated because there is no room for improvement of the schedule.

4.2.2 Modifiable Operations

A schedule in the dynamic scheduling contains three kinds of operations, the operations had already been processed, the operations in processing

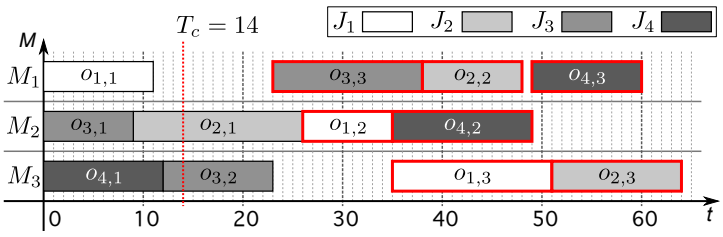


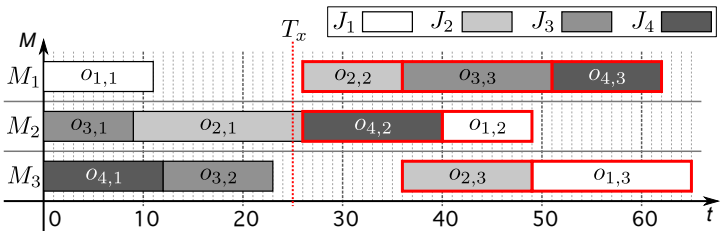
Figure 4.2: A modified schedule

and the operations not began to process yet. To modify the schedule, the RS process can modify the sequences of the operations not processed yet. In this chapter, these operations (not began to process yet) are called the *modifiable operations* and the others are called the *unmodifiable operations*. From the definition of the modifiable operations, the operations $\{o_{1,2}, o_{1,3}, o_{2,2}, o_{2,3}, o_{3,3}, o_{4,2}, o_{4,3}\}$ are modifiable and the others are unmodifiable in Figure 4.1(b). By rearranging the sequences of modifiable operations, the modified schedule shown in Figure 4.2 is obtained.

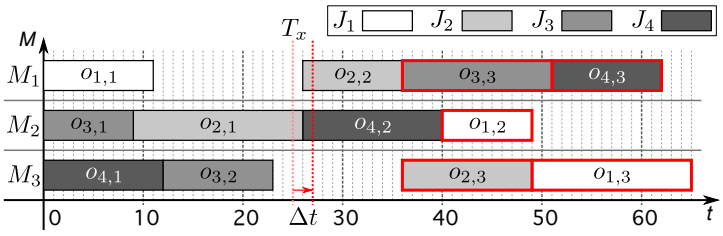
4.3 Application of LCO to RS

This section proposes an algorithm for the RS process using LCO. The following notations are used in the description of the proposed algorithm.

x	:	the number of steps in the RS process
T_x	:	the start time of the x^{th} step in the RS process
Δt	:	the given computation time for LCO
$S_c(x)$	:	the solution in the x^{th} step in LCO



(a) At time T_x



(b) At time $T_x + \Delta t$

Figure 4.3: Transition of the modifiable operations

The initial solution for the RS process is generated by listing the modifiable operations at the time $T_x|_{x=1}$. For example, in Figure 4.3(a), the operations boxed with bold red lines are modifiable and $S_c(x = 1)$ is generated as Equation 4.1 by listing the modifiable jobs ordered by their start time on the existing schedule.

$$S_c(1) := (J_2, J_4, J_3, J_2, J_1, J_1, J_4) \tag{4.1}$$

In addition, to deal with the dynamic environment, the RS process using LCO is required to consider the computation time in the algorithm. For instance, although the operations $o_{2,2}$ and $o_{4,2}$ are modifiable at a certain time T_x as shown in Figure 4.3(a), those operations become unmodifiable at $T_x + \Delta t$

as shown in Figure 4.3(b) because they are began to process during the computation. Based on the above considerations, this section proposes the algorithm of the RS process using LCO is shown as the following procedure.

step 1. Initialization

Set step $x \leftarrow 1$

step 2. Generate $S_c(x)$

If $x = 1$, which means that $T_x|_{x=1}$ is equal to the given time T_c, this step generates an initial solution $S_c(1)$ by extracting modifiable operations from the existing schedule. The modifiable operations satisfy the condition shown in Equation 4.2, where $s_{j,k}$ denotes the start time of the operation $o_{j,k}$.

$$s_{j,k} \geq T_x|_{x=1} \tag{4.2}$$

The sequence of jobs on the solution is determined by their start time on the existing schedule.

On the other hand, if $x \geq 2$, set $S_c(x) \leftarrow S_c(x-1)$ in order to inherit the existing solution.

step 3. Remove the unmodifiable operations from $S_c(x)$

$S_c(x)$ can contain the operations which become unmodifiable during the computation for the RS process. Such operations should be removed from $S_c(x)$ in terms of the efficiency of the optimization. This step removes the jobs correspond to the operations $o_{j,k}$ which satisfy

Equation 4.3 from $S_c(x)$.

$$s_{j,k} \quad < \quad T_x + \Delta t \tag{4.3}$$

The removal of the unmodifiable operations from the solution representation is performed by the following procedure.

1. Generate a set of unmodifiable operations U according to Equation 4.3.
2. Select an operation $o_{j,k} \in U$.
3. Select the job J_j corresponds to the operation $o_{j,k}$ from the solution representation using the liner search.
4. Remove the selected operation $o_{j,k}$ from U and the selected job J_j from the solution representation.
5. Iterate the processes from 2 to 4 until the set U is empty. Otherwise, terminate this procedure.

step 4. Optimization using LCO

This step applies the optimization performed by LCO to $S_c(x)$ for the calculation time Δt. The calculation time is simply defined as the elapsed time from T_x.

step 5. Update the schedule

This step replaces the current schedule for modifiable operations with the schedule decoded from $S_c(x)$.

step 6. Loop or terminate

If the termination condition is satisfied, terminate this procedure. Otherwise, go back to step 2 after replacing x with $x + 1$.

The fundamental framework of the above procedure is based on the conventional algorithm using GA proposed by Tanimizu et al. [TSS03]. The steps from 1 to 2 in the above procedure are almost the same processes as the conventional algorithm. On the other hand, the step 3 is one of particular processes in the RS process using LCO.

In RS process, the solution can provide unacceptable modifications of the schedule due to the unmodifiable operations, which often disrupts subsequent searches and causes serious inefficiency in searches. Such problem also happens in other metaheuristic searches, including GA. The method to avoid the problem is proposed in the conventional algorithm, however, it cannot be applied in our proposed method since LCO widely differs from GA in the searching mechanism. The step 3 in the procedure provides LCO with the mechanism to avoid the problem by removing the jobs correspond to the unmodifiable operations. It is guaranteed that the schedule given by the solution representation does not change before or after the step. In this study, the linear search is used to select and remove the unmodifiable operations, which is a different point from the conventional algorithm using GA.

The number of clustering iteration in the proposed algorithm depends on the definition of Δt at the step 4 in the procedure. If Δt is much longer than the calculation time of a clustering, LCO optimizes the current schedule

dramatically. Otherwise, if Δt is too short, LCO cannot work well. Also, GA which is used in the original study and other optimization algorithms have similar characteristics.

It is guaranteed that the cost of the schedule determined by $S_c(x)$ is equal to or better than the cost of the existing schedule because LCO is an algorithm based on the greedy search. Therefore, the schedule should not be worse at the step 5 in the procedure.

4.4 Numerical Experiments

This section verifies the effectiveness of the proposed algorithm with some numerical experiments. The performance of the proposed algorithm is discussed by comparing it with the conventional algorithm based on GA in terms of the modification capability and the computation time to modify the schedule.

4.4.1 Comparative Method : RS Process using GA

As a comparative method, this study uses RS process using GA proposed by Tanimizu et al. [TSS03]. Instead of the clustering, RS process using GA applies the genetic operations to modify the schedule.

In general, the genetic operations on GA consist of *crossover*, *mutation* and *selection*. The crossover is a mechanism to generate several offspring from several parent individuals. Although many crossover methods are pro-

posed for the scheduling problem, this study adopts the precedence preserving order-based crossover (POX) proposed by Lee et al. [LYL98] as the crossover method. The mutation is a mechanism to apply perturbation, little changes, to each individual. In this study, the mutation is performed by swapping two loci on the chromosome in each individual.

The selection performs the alternation of generations. There are many methods for the selection, which are called the roulette selection, the ranking selection and so on. In this study, we adopt the tournament selection as the selection method on GA. In order to preserve the best individual through the alternations of generations, we also implement the elite preservation strategy into the selection method.

4.4.2 Experimental conditions

To associate the processing time with the computation time, this study defines the unit of the processing time as seconds. For instance, $p_{j,k} = 10$ means that it takes 10 seconds to process the operation $o_{j,k}$. We also define Δt as 1 second, which is the minimum unit of makespan and processing time in the schedule.

Because there is no benchmark problem for the reactive scheduling, this study randomly generates eight instances. The generated instances are shown in Table 4.1, where the following notations are used.

Table 4.1: Instances for the experiments

instance	n	m	C_p	C_e	T_c
R1	50	10	18420	20515	1842
R2	50	10	18020	21693	1802
R3	50	10	17923	18628	1793
R4	50	10	17745	22190	1775
S1	20	20	11390	12692	1139
S2	20	20	10859	13874	1086
S3	20	20	9998	11926	1000
S4	20	20	10993	11445	1100

n : the number of jobs
m : the number of machines
C_p : the makespan of the predetermined schedule
C_e : the makespan of the changed schedule
T_c : the time to change the schedule

The first four instances denoted as R1, R2, R3 and R4 consist of 50 jobs and 10 machines. These instances are as large as the instances shown in [TSS03] and they have relatively weak multimodality as shown in Chapter 2 and 3. The last four instances denoted as S1, S2, S3 and S4 consists of 20 jobs and 20 machines, which have relatively strong multimodality. The makespan of predetermined schedule and that of changed schedule are also shown in Table 4.1. For each instance, the time T_c when the schedule is changed is determined as Equation 4.4, where $\lceil * \rceil$ means the ceiling function.

$$T_c = \lceil \alpha C_p \rceil \quad (0 < \alpha < 1;\ \alpha \in \mathbb{R}) \tag{4.4}$$

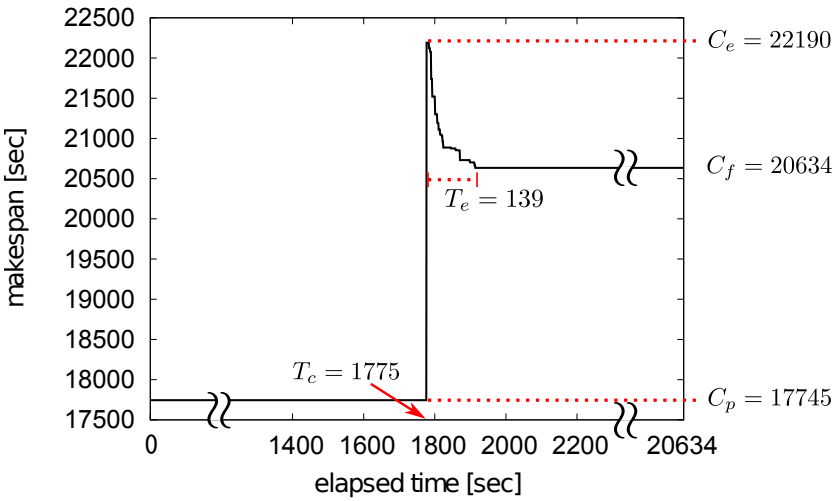


Figure 4.4: Transition of makespan and notations

In this study, the instances are created on the basis of the condition $\alpha = 0.10$ to give many modifiable operations.

In the experiments, the performance of the algorithms is evaluated in terms of the improvement capability and computation time required by the RS process. The improvement capability is measured by the relative error rate between the obtained makespan and the optimal makespan. Because the optimal makespan for each instance is unknown, this chapter uses the best known makespan for each instance in the experiments as the estimated optimal makespan. The relative error rate RE is calculated by Equation 4.5, where the notation C_f denotes the obtained makespan and C_b denotes the estimated optimal makespan.

$$RE = \frac{C_f - C_b}{C_b} \times 100 \tag{4.5}$$

Table 4.2: Parameters for each algorithm

(a) LCO

Parameter	Value
SEM : SIM : IEM	7 : 2 : 1
range of clustering radius	$\left[\frac{nm}{4}, \frac{nm}{3}\right]$

(b) GA

Parameter	Value
population	500
tournament size	50
crossover rate	0.70
mutation rate	0.10

In addition to the relative error rate, this study also discusses the computation time. The computation time T_e is evaluated by the elapsed time from the time when the schedule changed to the time when the obtained makespan C_f is obtained for the first time. Figure 4.4 briefly shows what each notation denotes in the transition of the makespan.

The parameters of LCO and GA for the RS process are shown in Table 4.2. In this study, the clustering radius on LCO is determined at random per clustering. In addition, because the initial schedule is determined by LCO, the parameters of LCO is also used in pre-scheduling.

4.4.3 Experimental Results

Firstly, we focus on the relative error rate. Figure 4.5 shows the distribution of the relative error rate RE for each method over 10 trial. From Figure 4.5, the RS process using LCO always indicates better performance than that using GA. This result suggests that LCO is more effective method for the

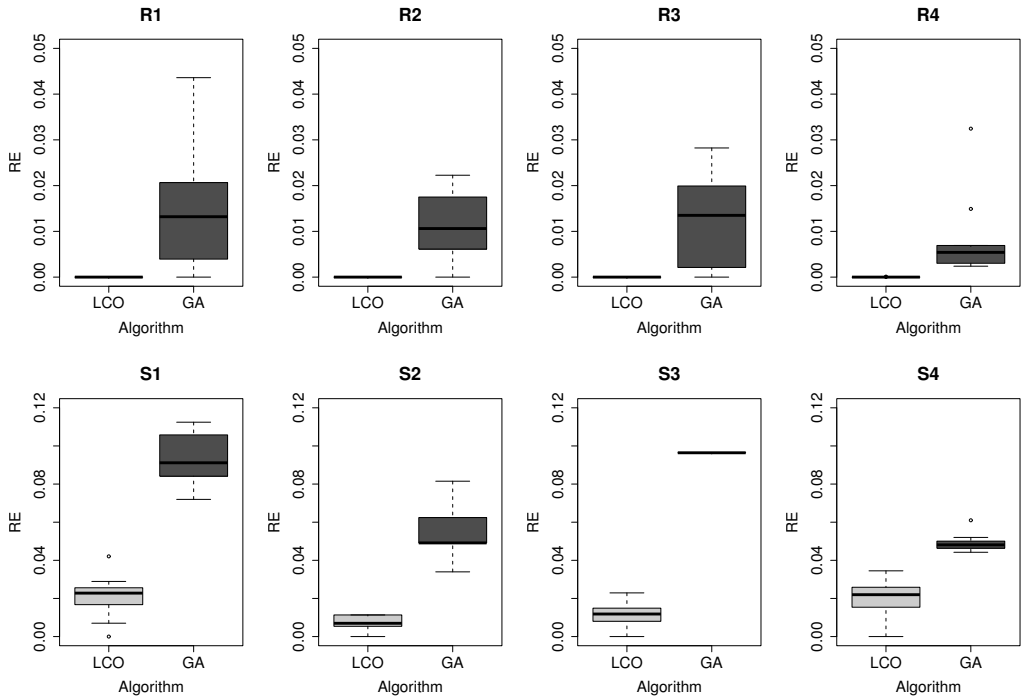


Figure 4.5: Experimental results: relative error rate to the best makespan

RS process than GA in terms of the modification capability of the schedule.

Secondly, we focus on the computation time. Figure 4.6 shows an example of the transition of the makespan in R4. The horizontal axis in Figure 4.6 denotes the elapsed time to process operations in accordance with a schedule in the factory. After the time $T_c = 1775$ [sec], the schedule is prolonged and the RS process improves the schedule every 1 second because Δt is set to 1 second. From Figure 4.6, the makespan of the schedule modified by the RS process using LCO decreases much faster than the one optimized by the RS process using GA. Although this trend of the computation time is also confirmed in other cases, to analyze the trends statistically, Figure 4.7

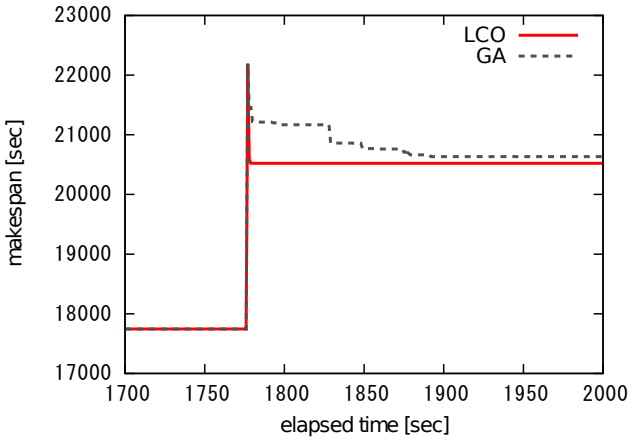


Figure 4.6: Experimental results: typical transition of makespan

shows the performance related to the computation time. From Figure 4.7, the computation time required by the RS process using LCO is significantly shorter than that required by the RS process using GA in almost all cases. In particular, for the instances R1–R4, though the RS process using GA requires several hundred steps (seconds) to modify the schedule at worst, the RS process using LCO requires just a few steps (seconds). On the other hand, the computation time required by the RS process using GA is significantly shorter than the one required by the proposed algorithm for the instance S3. This trend is far different from the trends shown by the other cases.

4.5 Discussion

From the characteristics or definitions of the RS, it is obvious that the shorter Δt gives more opportunities to improve the schedule and the effectiveness of

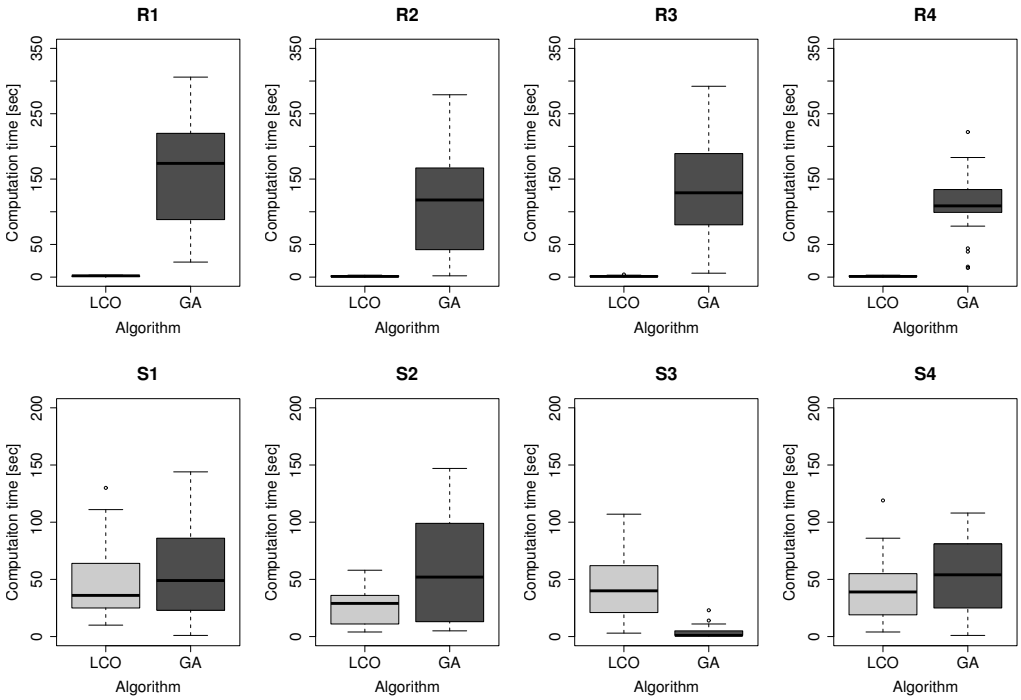


Figure 4.7: Experimental results: computation time

the RS process largely depends on the efficiency of the searching. Thus, it is considered that the RS process using LCO is more effective than that using GA because LCO is much better algorithm than GA in terms of the time-effectiveness. This section discusses the effectiveness of the proposed RS process using LCO on the basis of the experimental results shown in Figure 4.5 and 4.7.

The effectiveness of the proposed algorithm for the instances R1–R4 proves the above consideration. From the figures, especially for the instances R1–R4, the effectiveness of the proposed algorithm is obviously verified in terms of both the modification capability and the computation time. On the

other hand, because the proposed algorithm can obtain the modified schedule almost within a step (1 second), the proposed algorithm only optimizes the schedule as a static scheduling problem. The results for the instances S1–S4 are more interesting and important to analyze the effectiveness of the proposed algorithm for the dynamic scheduling.

From Figure 4.7, for the instances S1–S4, although the computation time of the proposed algorithm is shorter than the comparative algorithm in many cases, there is just a small difference. In addition, for the instance S3, the computation time of the proposed algorithm is longer than the comparative algorithm. On the other hand, from Figure 4.5, the effectiveness of the proposed algorithm for the instances S1–S4 in terms of the modification capability is still higher than that using GA significantly. These results suggest that GA has difficulties to find a better schedule than LCO and the proposed algorithm searches for the improved schedule as short computation time as possible.

Summarizing these discussion, the effectiveness of the proposed algorithm is based on both the fast computation and the high accuracy of LCO. Also, the effectiveness of the proposed algorithm is proved in both the strong multimodal problems and the weak multimodal problems.

4.6 Conclusion

This chapter proposes a new solution method based on LCO for the reactive scheduling. In the proposed method, LCO is provided with the mechanism to deal with the solution representation which dynamically changes throughout the RS process. The effectiveness of the proposed method is discussed as compared to the conventional algorithm using genetic algorithm in a numerical experiment.

According to the numerical experiment, the effectiveness of the proposed method is summarized as follows;

- The modification capability rate of the proposed method is averagely better than the comparative algorithm using GA.
- The proposed method can also improve the schedule faster than GA.
- The numerical experiment obviously suggests that the RS process using LCO is more effective method than the one using GA.

There is a room to improve the performance of GA which is a comparative method in the experiment. In this paper, although GA generally has higher concurrency than LCO, GA is not implemented for parallel computing as well as LCO. The implementation of GA for parallel computing and the experiment using it is the future work.

Bibliography

[LYL98] Kyung-Mi Lee, T. Yamakawa, and Keon-Myung Lee. A genetic algorithm for general machine scheduling problems. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on*, volume 2, pages pp. 60–66, 1998.

[TIYF14] Yasumasa Tamura, Hiroyuki Iizuka, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization to reactive job-shop scheduling. *Soft Computing*, pages pp. 1–9, 2014. (Published online).

[TSS03] Yoshitaka Tanimizu, Tatsuhiko Sakaguchi, and Nobuhiro Sugimura. Genetic algorithm based reactive scheduling : 1st report, modification of production schedule for delays of manufacturing processes. *Transactions of Japan Society of Mechanical Engineers C*, 69(685):pp. 2458–2463, 2003. (in Japanese).

[TYF13] Yasumasa Tamura, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization to reactive job-shop scheduling. In *Proceedings of The 14th International Symposium on Advanced Intelligent Systems (ISIS2013)*, pages T1f–4, 2013.

Chapter 5

Conclusion

In Chapter1, this thesis introduced the research background and objective. After the description of the mathematical model, the feasible region and the solution representations for the job-shop scheduling problem (JSP), we introduced the conventional solution algorithms employing local search. The advantages and disadvantages of the conventional algorithms are also discussed to analyze their effectiveness.

In Chapter2, we proposed the hybrid algorithm based on local clustering organization (LCO) and simulated annealing (SA) for the static JSP. To improve the performance of LCO for the instances which have relatively strong multimodality, the mechanism proposed in Chapter 2 is based on SA to apply the random effect to the control mechanism for the transition of the neighborhood solutions. The effectiveness of the proposed algorithm is verified by comparing it with the conventional LCO and the conventional SA

which is one of the most effective algorithm for the JSP.

In Chapter 3, we discuss the mechanism to escape from local optima for the instances which have relatively weak multimodality. When the fitness landscape has weak multimodality, the distance between the local optima tends to be long. Therefore, we proposed the mechanism to realize the long-jump in the solution space. In addition, we also proposed the novel neighborhood search algorithm based on the priority rules, the practical heuristics for the JSP. The effectiveness of the proposed algorithm is verified by comparing it with the conventional LCO and the conventional SA as well as Chapter 2. Chapter 3 also proves the relation between the effective mechanism to escape from local optima and the multimodality of the instances by comparing the algorithms proposed in Chapter 2 and Chapter 3.

In Chapter 4, we dealt with the algorithm for the dynamic scheduling. The reactive scheduling is a methodology to modify the predetermined schedule on the basis of the variable environment in the manufacturing systems without suspending the production line. We proposed the reactive scheduling process using LCO and the effectiveness of the proposed algorithm is discussed by comparing it with the conventional algorithm based on genetic algorithm. As a result, the proposed algorithm had higher performance for modifying the schedule than the conventional algorithm.

Finally, we summarize this thesis with some remarks. This thesis deals with the metaheuristics for the job-shop scheduling problem. The studies in the thesis propose the effective metaheuristics employing local clustering for

both static and dynamic scheduling problems.

For static scheduling, this thesis proposed two individual approaches by considering the strength of the multimodality. The one is aimed at solving the instances which have strong multimodality effectively, and the other is aimed at solving the instances which have relatively weak multimodality. According to the studies for the static scheduling, this thesis proves that the effective mechanism to escape from local optima should be decided depending on the multimodality of the target instances as follows.

- When the target instance has strong multimodality, the mechanism to escape from local optima should be based on the probabilistic acceptance of worse transition of the solution like the simulated annealing because the distance between local optima tends to be close in such a instance.

- When the target instance has relatively weak multimodality, the mechanism should be based on the large-scale neighborhood search because the distance between local optima tends to be far.

The effectiveness of the following methods are also proved in the studies.

- Local search method using permutation with repetition

- Neighborhood decision based on the heuristics for the large-scale neighborhood search

The latter fact is a significant contribution to the general design of meta-heuristics for the combinatorial optimization.

For the dynamic scheduling, this thesis proposed the reactive scheduling method using local clustering organization. The high effectiveness of the proposed method was proved in terms of both the modification capability and the computation time. It is also a contribution to the production scheduling in the real manufacturing systems.

Acknowledgement

The author expresses his heartfelt gratitude to Professor Masahito Yamamoto of Hokkaido University for constant polite instruction and guidance from the inceptions to the completion of the present research.

Deep thanks are also due to Professor Masahito Kurihara, Professor Tetsuo Ono, Professor Keiji Suzuki and Associate Professor Hiroyuki Iizuka of Hokkaido University.

He would like to thank Professor Masashi Furukawa of Hokkaido Information University and Associate Professor Ikuo Suzuki of Kitami Institute of Technology for profitable discussion and suggestions.

Many pieces of affectionate advice and discussion given by them were very useful to improve this research.

He would like to thank all students and staff of the office, Autonomous Systems Engineering Laboratory, Synergetic Information Engineering. The conversations between them, particularly Soichiro Yokoyama, Jun Ogawa, Masaya Honjo, Zhuoran Xu and Yukihiro Tagawa, always encouraged him.

Finally, the author would like to thank to his family; his father Yoshimi

Tamura, his mother Akemi Tamura and his brother Takahiro Tamura. Without their support, he would not have continued his research.

This study was also supported by JSPS KAKENHI, Grant-in-Aid for JSPS Fellows, 26·1342.

January 28, 2015

Yasumasa Tamura

Bibliography

[ABZ88] Joseph Adams, Egon Balas, and Daniel Zawack. The shifting bottleneck procedure for job shop scheduling. *Manage. Sci.*, 34(3):pp. 391–401, March 1988.

[Bak74] K.R. Baker. *Introduction to sequencing and scheduling.* Wiley, 1974.

[Bal69] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations Research*, 17(6):pp. 941–957, 1969.

[BDP96] Jacek Blazewicz, Wolfgang Domschke, and Erwin Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):pp. 1–33, 1996.

[Bie95] Christian Bierwirth. A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, 17(2-3):pp. 87–92, 1995.

[BV98] Egon Balas and Alkis Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, 44(2):pp. 262–275, 1998.

[BW69] G.H. Brooks and C.R. White. An algorithm for finding optimal or near optimal solutions to the production scheduling problem. *The Journal of Industrial Engineering*, 16(1):pp. 34–40, 1969.

[CB76] E.G. Coffman and J.L. Bruno. *Computer and job-shop scheduling theory*. A Wiley-Interscience publication. Wiley, 1976.

[CMM67] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of scheduling*. Addison-Wesley Pub. Co., 1967.

[DT93] Mauro Dell'Amico and Marco Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41(3):pp. 231–252, 1993.

[FMW05] Masashi Furukawa, Yusuke Matsumura, and Michiko Watanabe. Local clustering organization (LCO) solving a large scale of tsp. *Journal of Robotics and Mechatronics*, 17(5):pp. 560–567, 2005. (in Japanese).

[FMW06] Masachi Furukawa, Yusuke Matsumura, and Michiko Watanabe. Development of local clustering organization applied to job-shop scheduling problem. *Journal of the Japan Society for Precision Engineering (CD-ROM)*, 72(7):pp. 867–872, 2006. (in Japanese).

[Fre82] S. French. *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. Ellis Horwood series in mathematics and its applications. E. Horwood, 1982.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[GJS76] Micheal R. Garey, David S. Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:pp. 117–129, 1976.

[GT60] B. Giffler and G. L. Thompson. Algorithms for solving production-scheduling problems. *Operations Research*, 8(4):487–503, 1960.

[JM99] A.S. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):pp. 390–434, 1999.

[Koh98] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1–3):pp. 1–6, 1998.

[KS09] Yohko Konno and Keiji Suzuki. Performance of extended local clustering organization (LCO) for large scale job-shop scheduling problem (JSP). *IEEJ Transactions on Electronics Information and Systems*, 129(7):pp. 1363–1370, 2009. (in Japanese).

[LAL92] Peter J. M. van Laarhoven, Emile H. L. Aarts, and Jan Karel Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):pp. 113–125, 1992.

[Law84] S. Lawrence. Resource constrained project scheduling: An experimental investigation of heuristics scheduling techniques. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1984.

[LYL98] Kyung-Mi Lee, T. Yamakawa, and Keon-Myung Lee. A genetic algorithm for general machine scheduling problems. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on*, volume 2, pages pp. 60–66, 1998.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):pp. 1087–1092, 1953.

[NS96] Eugeniusz Nowicki and Czeslaw Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):pp. 797–813, 1996.

[NS05] Eugeniusz Nowicki and Czeslaw Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*,

8(2):145–159, 2005.

[PI77] S. S. Panwalkar and Wafik Iskander. A survey of scheduling rules. *Operations Research*, 25(1):pp. 45–61, 1977.

[PM00] Ferdinando Pezzella and Emanuela Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120(2):pp. 297–310, 2000.

[RK76] A.H.G. Rinnooy Kan. *Machine scheduling problems: classification, complexity and computations*. Nijhoff, 1976.

[SWV92] Robert H. Storer, S. David Wu, and Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495–1509, 1992.

[Tai94] E.D. Taillard. Parallel taboo search techniques for the job shop scheduling problem. *ORSA Journal on Computing*, 6(2):pp. 108–117, 1994.

[TIY15] Yasumasa Tamura, Hiroyuki Iizuka, and Masahito Yamamoto. Extended local clustering organization with rule-based neighborhood search for job-shop scheduling problem. In *Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems - Volume 2*, volume 2 of *Proceedings in Adaptation,*

Learning and Optimization, pages pp. 465–477. Springer International Publishing, 2015.

[TIYF14] Yasumasa Tamura, Hiroyuki Iizuka, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization to reactive job-shop scheduling. *Soft Computing*, pages pp. 1–9, 2014. (Published online).

[TSS03] Yoshitaka Tanimizu, Tatsuhiko Sakaguchi, and Nobuhiro Sugimura. Genetic algorithm based reactive scheduling : 1st report, modification of production schedule for delays of manufacturing processes. *Transactions of Japan Society of Mechanical Engineers C*, 69(685):pp. 2458–2463, 2003. (in Japanese).

[TSYF12] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. In *Proceedings of ASME/ISCIE 2012 International Symposium on Flexible Automation (ISFA2012)*, pages ISFA2012–7226, 2012.

[TSYF13] Yasumasa Tamura, Ikuo Suzuki, Masahito Yamamoto, and Masashi Furukawa. The hybrid approach of lco and sa to solve job-shop scheduling problem. *Transactions of the Institute of Systems, Control and Information Engineers*, 26(4):pp. 121–128, 2013.

[TYF13] Yasumasa Tamura, Masahito Yamamoto, and Masashi Furukawa. Application of local clustering organization to reactive job-shop scheduling. In *Proceedings of The 14th International Symposium on Advanced Intelligent Systems (ISIS2013)*, pages T1f–4, 2013.

[TYSF13] Yasumasa Tamura, Masahito Yamamoto, Ikuo Suzuki, and Masashi Furukawa. Acquisition of dispatching rules for job-shop scheduling problem by artificial neural networks using pso. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 17(5):pp. 731–738, 2013.

[WHW05] Jean-paul Watson, Adele E. Howe, and L. Darrell Whitley. Deconstructing nowicki and smutnicki's i-tsab tabu search algorithm for the job-shop scheduling problem. *Computers & Operations Research*, 33:pp. 2623–2644, 2005.

[ZF94] M. Zweben and M.S. Fox. *Intelligent Scheduling.* Morgan Kaufmann Publishers, 1994.

[ZLGR07] Chao Yong Zhang, Pei Gen Li, Zai Lin Guan, and Yun Qing Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, 34(11):pp. 3229–3242, 2007.